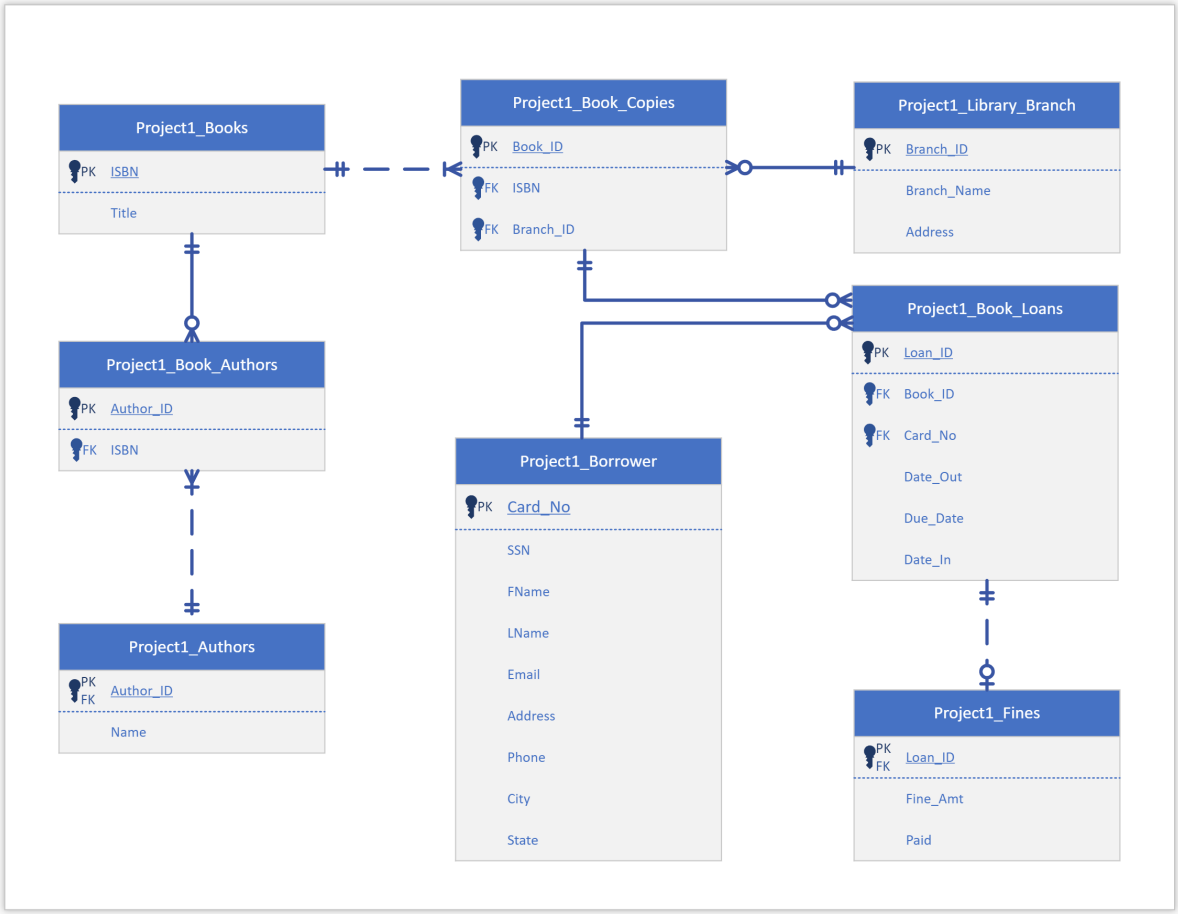


TABLE CREATION WITH KEY DESIGNATIONS

CREATE TABLE Project1_Books (
ISBN VARCHAR2(10) NOT NULL, --!!!!!Choosing to go with the ISBN10 number
Title VARCHAR2(256) NOT NULL,

```
CONSTRAINT BookPK PRIMARY KEY(ISBN)
```

);

CREATE TABLE Project1_Book_Authors(
Author_ID VARCHAR2 (13) NOT NULL,
ISBN VARCHAR2 (10) NOT NULL,

```
CONSTRAINT BKAuthorPK   PRIMARY KEY (Author_ID),
CONSTRAINT BKAuthorFK   FOREIGN KEY (ISBN)
    REFERENCES Project1_Books(ISBN)
);
```

CREATE TABLE Project1_Authors(
Author_ID VARCHAR2(13) NOT NULL,
Name VARCHAR2(100) NOT NULL,

```
CONSTRAINT  AuthorNamesPK   PRIMARY KEY (Author_ID),
CONSTRAINT  AuthorNamesFK   FOREIGN KEY (Author_ID)
    REFERENCES Project1_Book_Authors(Author_ID)
);
```

CREATE TABLE Project1_Library_Branch (
Branch_ID NUMBER (2) NOT NULL,
Branch_Name VARCHAR2(30) NOT NULL,
Address VARCHAR2(50) NOT NULL,

```
CONSTRAINT BranchPK PRIMARY KEY (Branch_ID)
);
```

CREATE TABLE Project1_Book_Copies(
Book_ID VARCHAR2(20) NOT NULL,
ISBN VARCHAR2(10) NOT NULL,
Branch_ID NUMBER(2) NOT NULL,

```
CONSTRAINT Book_CopiesPK        PRIMARY KEY (Book_ID),
CONSTRAINT Book_Copies_ISBN_FK  FOREIGN KEY (ISBN)
    REFERENCES Project1_Books(ISBN),
CONSTRAINT Book_Copies_BrID_FK  FOREIGN KEY (Branch_ID)
    REFERENCES Project1_Library_Branch(Branch_ID)
);
```

CREATE TABLE Project1_Borrower (
Card_No VARCHAR2(10) NOT NULL,
SSN VARCHAR2(11) NOT NULL UNIQUE,
FName VARCHAR2(20) NOT NULL,
LName VARCHAR2(30) NOT NULL,
Email VARCHAR2(50) NOT NULL, --Going of Assumption that you must provide some sort of contact email
Address VARCHAR2(50) NOT NULL, --Going of Assumption that you must provide some sort of contact address
City VARCHAR2(30) NOT NULL,
State VARCHAR2(2) NOT NULL,
Phone VARCHAR2(14) NOT NULL, --Going of Assumption that you must provide some sort of contact #

```
CONSTRAINT BorrowerPK         PRIMARY KEY (Card_NO)
);
```

CREATE TABLE Project1_Book_Loans(
Loan_ID VarChar2(40) NOT NULL, -- Will need to generate this from the sequence and follow Rules
Book_ID VARCHAR2(20) NOT NULL, -- Length will depend on how Book_ID generated previously
Card_no VARCHAR2(10) NOT NULL, -- Pull from Borrower ID0000ID
Date_Out DATE NOT NULL, -- No rules given for date requirements
Due_Date DATE NOT NULL, -- No rules given for date requirements.
Date_in DATE NULL, -- 50 should be after Due_Date, This will feed into the Fines Table Query

```
CONSTRAINT  Book_LoanPK       PRIMARY KEY (Loan_ID),
CONSTRAINT  Book_LoanFK1      FOREIGN KEY (Book_ID)
    REFERENCES   PROJECT1_Book_Copies (Book_ID),
CONSTRAINT  Book_LoanFK2      FOREIGN KEY (Card_No)
    REFERENCES   PROJECT1_Borrower (Card_No)
);
```

CREATE TABLE Project1_Fines(
Loan_ID VARCHAR2(40) NOT NULL, --Pull from Loan_ID in
Fine_Amt NUMBER (4,2) NOT NULL, --"Fine amount is $0.25 per day late"
PAID VARCHAR2(6) NOT NULL, --Indicate if the Fine is paid (No specification in project guidelines)

```
CONSTRAINT FinesPK PRIMARY KEY (Loan_ID),
CONSTRAINT FinesFK FOREIGN KEY (Loan_ID)
    REFERENCES Project1_Book_Loans(Loan_ID)
);
```

DATA LOAD, NORMALIZATION, DATA GENERATION

Since my Create Table SQL are generated throughout my SQL code, I will leave it in the section below.

```
--Rob Lindsay
--RXL200006
--BUAN 6320.001
--Project 1
--*****************
--*****************
--Create Book Table with ISBN, Title

DROP TABLE Project1_Books;
CREATE TABLE Project1_Books (
   ISBN VARCHAR2(10) NOT NULL, --!!!!!Choosing to go with the ISBN10 number
   Title  VARCHAR2(256) NOT NULL,
```

```
CONSTRAINT BookPK PRIMARY KEY(ISBN)
```

```
);

INSERT INTO Project1_Books
      SELECT ISBN10 , Title
      FROM project1_books_load;

--SELECT * FROM Project1_Books;

--*****************
--*****************
--Create Book_Authors Table with Aurthor_ID, ISBN
    --!!!!Unique ID could be ISBN || Author # of book since our system is not
    --     designed to have an author write multiple books
--First Create Temp_Author_Table to contain ISBN, Author, CommaCount(Author)
DROP TABLE Temp_Author_Table;
CREATE TABLE Temp_Author_Table AS (
   SELECT  ISBN10, Author,
         LENGTH(REPLACE(Author,',','||')) - LENGTH (Author)+1 AS Number_Authors,
         regexp_substr(Author, '[^,]+' , 1 , 1) AS Author1,
         ISBN10 || '1' AS Author1ID,
         regexp_substr(Author, '[^,]+' , 1 , 2) AS Author2,
         ISBN10 || '2' AS Author2ID,
         regexp_substr(Author, '[^,]+' , 1 , 3) AS Author3,
         ISBN10 || '3' AS Author3ID,
         regexp_substr(Author, '[^,]+' , 1 , 4) AS Author4,
         ISBN10 || '4' AS Author4ID,
         regexp_substr(Author, '[^,]+' , 1 , 5) AS Author5,
         ISBN10 || '5' AS Author5ID
   FROM PROJECT1_Books_Load)
   ;
--SELECT * FROM Temp_Author_Table ORDER BY Number_Authors Desc;
--DESC TEMP_Author_Table;
```

--Now Create Book_Authors Table with Author_ID, ISBN
DROP TABLE Project1_Book_Authors;
CREATE TABLE Project1_Book_Authors(
Author_ID VARCHAR2 (13) NOT NULL,
ISBN VARCHAR2 (10) NOT NULL,

```
CONSTRAINT BKAuthorPK   PRIMARY KEY (Author_ID),
CONSTRAINT BKAuthorFK   FOREIGN KEY (ISBN)
    REFERENCES Project1_Books(ISBN)
);
```

--Insert each of the AuthorID columns and their respective ISBN into Book_Authors
INSERT INTO Project1_Book_Authors
SELECT Author1ID, ISBN10 FROM Temp_Author_Table WHERE Author1 is not null;
INSERT INTO Project1_Book_Authors
SELECT Author2ID, ISBN10 FROM Temp_Author_Table WHERE Author2 is not null;
INSERT INTO Project1_Book_Authors
SELECT Author3ID, ISBN10 FROM Temp_Author_Table WHERE Author3 is not null;
INSERT INTO Project1_Book_Authors
SELECT Author4ID, ISBN10 FROM Temp_Author_Table WHERE Author4 is not null;
INSERT INTO Project1_Book_Authors
SELECT Author5ID, ISBN10 FROM Temp_Author_Table WHERE Author5 is not null;
--SELECT * FROM Project1_Book_Authors;

--**********
--**********
--Create Authors Table with AuthorID, Name
--!!Still Need To Do: No Author Given Scenario, everything else is done
--!!!!USE The Same Code As The Book_Authors but with Name instead of ISBN
--!!!!Could also create a temporary table of ISBN, AuthorName, AuthorID the Drop temp table

DROP TABLE Project1_Authors;
CREATE TABLE Project1_Authors(
Author_ID VARCHAR2(13) NOT NULL,
Name VARCHAR2(100) NOT NULL,

```
CONSTRAINT  AuthorNamesPK   PRIMARY KEY (Author_ID),
CONSTRAINT  AuthorNamesFK   FOREIGN KEY (Author_ID)
    REFERENCES Project1_Book_Authors(Author_ID)
);
```

--DELETE FROM Project1_Authors;

--Insert each of the AuthorID columns and their respective ISBN into Book_Authors
INSERT INTO Project1_Authors
SELECT Author1ID, Author1 FROM Temp_Author_Table WHERE Author1 is not null;
INSERT INTO Project1_Authors
SELECT Author2ID, Author2 FROM Temp_Author_Table WHERE Author2 is not null;
INSERT INTO Project1_Authors
SELECT Author3ID, Author3 FROM Temp_Author_Table WHERE Author3 is not null;
INSERT INTO Project1_Authors

```
    SELECT Author4ID, Author4 FROM Temp_Author_Table WHERE Author4 is not null;
INSERT INTO Project1_Authors
    SELECT Author5ID, Author5 FROM Temp_Author_Table WHERE Author5 is not null;
--SELECT * FROM Project1_Authors;

DROP TABLE Temp_Author_Table;        --This should be at the end of this section, created in
previous section

--*****************
--*****************
--Create Library_Branch Table with Branch_ID, Branch_Name, Address
CREATE TABLE Project1_Library_Branch (
  Branch_ID   NUMBER (2)      NOT NULL,
  Branch_Name VARCHAR2(30)    NOT NULL,
  Address     VARCHAR2(50)    NOT NULL,
```

```
CONSTRAINT BranchPK PRIMARY KEY (Branch_ID)
);
```

```
INSERT INTO Project1_Library_Branch
  SELECT Branch_ID, Branch_Name, Address
  FROM project1_library_branch_load
  ;
--SELECT * FROM Project1_Library_Branch;
COMMIT;

--*****************
--*****************
--Create Book_Copies Table with Book_ID, ISBN, Branch_ID
--!!!!Create a unique number by concatenating the BranchID, BookID, and CopyNo
--SELECT * FROM Project1_Book_Copies_Load;
--SELECT MAX(No_Of_Copies) from Project1_Book_Copies_Load;  --> 18 is the maximum number
of copies of a single book at a single branch

--FIRST, create a temporary copies table that creates the maximum possible amount of books (up
to 18) provided there is a copy at the branch
DROP TABLE Temp_Copies_Table;
CREATE TABLE Temp_Copies_Table AS (
  SELECT  Book_ID, Branch_ID, No_Of_Copies,
        Branch_ID || Book_ID || '01OF' ||No_OF_Copies AS Copy1,
        Branch_ID || Book_ID || '02OF' ||No_OF_Copies AS Copy2,
        Branch_ID || Book_ID || '03OF' ||No_OF_Copies AS Copy3,
        Branch_ID || Book_ID || '04OF' ||No_OF_Copies AS Copy4,
        Branch_ID || Book_ID || '05OF' ||No_OF_Copies AS Copy5,
        Branch_ID || Book_ID || '06OF' ||No_OF_Copies AS Copy6,
        Branch_ID || Book_ID || '07OF' ||No_OF_Copies AS Copy7,
        Branch_ID || Book_ID || '08OF' ||No_OF_Copies AS Copy8,
        Branch_ID || Book_ID || '09OF' ||No_OF_Copies AS Copy9,
        Branch_ID || Book_ID || '10OF' ||No_OF_Copies AS Copy10,
        Branch_ID || Book_ID || '11OF' ||No_OF_Copies AS Copy11,
        Branch_ID || Book_ID || '12OF' ||No_OF_Copies AS Copy12,
        Branch_ID || Book_ID || '13OF' ||No_OF_Copies AS Copy13,
        Branch_ID || Book_ID || '14OF' ||No_OF_Copies AS Copy14,
```

Branch_ID || Book_ID || '15OF' ||No_OF_Copies AS Copy15,
Branch_ID || Book_ID || '16OF' ||No_OF_Copies AS Copy16,
Branch_ID || Book_ID || '17OF' ||No_OF_Copies AS Copy17,
Branch_ID || Book_ID || '18OF' ||No_OF_Copies AS Copy18

```
FROM PROJECT1_Book_Copies_Load WHERE No_OF_Copies>0)
;
```

--SELECT * FROM Temp_Copies_Table;
--DESC Temp_Copies_Table;

--SECOND Create the Book_Copies Table
DROP TABLE Project1_Book_Copies;
CREATE TABLE Project1_Book_Copies(
Book_ID VARCHAR2(20) NOT NULL,
ISBN VARCHAR2(10) NOT NULL,
Branch_ID NUMBER(2) NOT NULL,

```
CONSTRAINT Book_CopiesPK        PRIMARY KEY (Book_ID),
CONSTRAINT Book_Copies_ISBN_FK  FOREIGN KEY (ISBN)
    REFERENCES Project1_Books(ISBN),
CONSTRAINT Book_Copies_BrID_FK  FOREIGN KEY (Branch_ID)
    REFERENCES Project1_Library_Branch(Branch_ID)
);
```

--THIRD Insert each of the Book_ID, ISBN and Branch ID whenever there are at least X copies in the branch
INSERT INTO Project1_Book_Copies
SELECT Copy1, Book_ID, Branch_ID FROM Temp_Copies_Table ;
INSERT INTO Project1_Book_Copies
SELECT Copy2, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>1;
INSERT INTO Project1_Book_Copies
SELECT Copy3, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>2;
INSERT INTO Project1_Book_Copies
SELECT Copy4, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>3;
INSERT INTO Project1_Book_Copies
SELECT Copy5, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>4;
INSERT INTO Project1_Book_Copies
SELECT Copy6, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>5;
INSERT INTO Project1_Book_Copies
SELECT Copy7, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>6;
INSERT INTO Project1_Book_Copies
SELECT Copy8, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>7;
INSERT INTO Project1_Book_Copies
SELECT Copy9, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>8;
INSERT INTO Project1_Book_Copies
SELECT Copy10, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>9;
INSERT INTO Project1_Book_Copies
SELECT Copy11, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>10;
INSERT INTO Project1_Book_Copies

SELECT Copy12, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>11;
INSERT INTO Project1_Book_Copies
SELECT Copy13, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>12;
INSERT INTO Project1_Book_Copies
SELECT Copy14, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>13;
INSERT INTO Project1_Book_Copies
SELECT Copy15, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>14;
INSERT INTO Project1_Book_Copies
SELECT Copy16, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>15;
INSERT INTO Project1_Book_Copies
SELECT Copy17, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>16;
INSERT INTO Project1_Book_Copies
SELECT Copy18, Book_ID, Branch_ID FROM Temp_Copies_Table WHERE No_Of_Copies>17;

--SELECT * FROM Project1_Book_Copies ORDER BY ISBN, Book_ID;

DROP TABLE Temp_Copies_Table; --This should be at the end of this section, created in previous section

--**********
--**********
--Create Borrower Table with Card_No, SSN, FName, LName, Address, Phone
DROP TABLE Project1_Borrower;
CREATE TABLE Project1_Borrower (
Card_No VARCHAR2(10) NOT NULL,
SSN VARCHAR2(11) NOT NULL UNIQUE,
FName VARCHAR2(20) NOT NULL,
LName VARCHAR2(30) NOT NULL,
Email VARCHAR2(50) NOT NULL, --Going of Assumption that you must provide some sort of contact email
Address VARCHAR2(50) NOT NULL, --Going of Assumption that you must provide some sort of contact address
City VARCHAR2(30) NOT NULL,
State VARCHAR2(2) NOT NULL,
Phone VARCHAR2(14) NOT NULL, --Going of Assumption that you must provide some sort of contact #

```
CONSTRAINT BorrowerPK        PRIMARY KEY (Card_NO)
);
```

--DESC Project1_Borrower;
select * from Project1_Borrowers_Load;
INSERT INTO Project1_Borrower
SELECT ID0000ID, SSN, First_Name, Last_Name, Email, Address, City, State, Phone
FROM Project1_Borrowers_Load;

--SELECT * FROM Project1_Borrower;
--**********
--Create Book_Loans Table with Loan_ID, Book_ID, Card_no, Date_Out, Due_Date, Date_in
--!!!! RULES FOR LOANS ASSIGNMENT
-- EXACTLY 400 books check-outs
-- FOR EXACTLY 200 different borrowers

--	AND Exactly 100 Different Books.
--	Same borrower should not check out same book more than once

DROP TABLE Project1_Book_Loans;
CREATE TABLE Project1_Book_Loans(
   Loan_ID   VarChar2(40)   NOT NULL,   -- Will need to generate this from the sequence and follow Rules
   Book_ID   VARCHAR2(20)   NOT NULL,   -- Length will depend on how Book_ID generated previously
   Card_no   VARCHAR2(10)   NOT NULL,   -- Pull from Borrower ID0000ID
   Date_Out   DATE   NOT NULL,   -- No rules given for date requirements
   Due_Date   DATE   NOT NULL,   -- No rules given for date requirements.
   Date_in   DATE   NULL,   -- 50 should be after Due_Date, This will feed into the Fines Table Query

```
CONSTRAINT  Book_LoanPK      PRIMARY KEY (Loan_ID),
CONSTRAINT  Book_LoanFK1     FOREIGN KEY (Book_ID)
    REFERENCES   PROJECT1_Book_Copies (Book_ID),
CONSTRAINT  Book_LoanFK2     FOREIGN KEY (Card_No)
    REFERENCES   PROJECT1_Borrower (Card_No)
);
```

DELETE Project1_Book_Loans;
INSERT INTO Project1_Book_Loans
 SELECT Card_No||Book_ID || Date_OUT AS Loan_ID,
      Book_ID, Card_No, Date_Out, Due_Date, Date_In
   FROM(
      SELECT Book_ID, Card_No,Book_Rank,Borrower_Rank, Lending_Period, Date_Out,
           Date_Out + 31 AS Due_Date,
           Date_Out + ROUND(DBMS_RANDOM.value(0,30),0) AS Date_IN
         FROM(SELECT Book_ID, Card_No,Book_Rank,Borrower_Rank, Lending_Period,
           TO_DATE('01-OCT-2021') - (93 * Lending_Period) + ROUND(DBMS_RANDOM.value(0,30),0) AS Date_Out
            FROM(
            SELECT Book_ID, Card_No,Book_Rank,Borrower_Rank, ROW_NUMBER() OVER (PARTITION BY Book_ID ORDER BY Borrower_Rank) AS Lending_Period
               FROM
               (SELECT * FROM(
                  SELECT Book_ID, Card_No,
                     DENSE_RANK() OVER (ORDER BY BOOK_ID) as Book_Rank,
                     ROW_NUMBER() OVER (PARTITION BY Book_ID ORDER BY Card_No) as Borrower_Rank
                     FROM (SELECT *
                        FROM (SELECT * FROM (SELECT BK.book_id FROM Project1_Book_Copies BK ORDER BY DBMS_RANDOM.RANDOM) WHERE rownum<=100) Book,
                           (SELECT * FROM (SELECT BO.Card_No FROM Project1_Borrower BO ORDER BY DBMS_RANDOM.RANDOM) WHERE rownum<=200) Borrower
)
)
                  WHERE Book_Rank = Borrower_Rank-1
                     OR Book_Rank = Borrower_Rank
                     OR Book_Rank = Borrower_Rank-101
                     OR Book_Rank = Borrower_Rank-100

```
                    OR (Book_Rank = 100 AND Borrower_Rank=1)
                ORDER By Book_ID,Borrower_Rank
            )
        )
    )
  );
--Now, Update the Loans so that 50 loans are returned after their due date
UPDATE Project1_Book_Loans SET Date_In = DUE_DATE + ROUND(DBMS_RANDOM.value(1,30),0)
WHERE rownum<51;
--SELECT * FROM Project1_Book_Loans;

--Now, select 10 loans that were not just updated for late return and adjust them to recently
checked out and not returned
UPDATE Project1_Book_Loans
  SET  Date_Out = SysDate - ROUND(DBMS_RANDOM.value(1,15),0) ,
    Date_In = NULL
  WHERE Loan_ID IN(
    SELECT Loan_ID FROM(
      SELECT  Loan_ID, Book_ID ,
        rank() over (partition by Book_ID order by Date_Out desc ) AS RANKING
      FROM Project1_Book_Loans
      WHERE Date_In<Due_Date AND Date_In IS NOT NULL Order BY Book_ID
      )
    WHERE Ranking = 1 and rownum < 11
    );
UPDATE Project1_Book_Loans SET  Due_Date = Date_Out+31 WHERE Date_IN IS NULL;
--SELECT * FROM Project1_Book_Loans WHERE Date_In IS NULL;

--SELECT * FROM Project1_Book_Loans WHERE Date_In IS NULL;
--SELECT * FROM Project1_Book_Loans;
--SELECT * FROM Project1_Book_Loans WHERE Due_Date < Date_In;

--*****************
--Create Fines Table with Loan_ID, Fine_Amt, Paid
--!!!! Will need to create SQL code to generate
--        EXACTLY 50 fines records for 50 DIFFERENT Borrowers
--        FINES Should be Generated by books checked back in late.
DROP TABLE Project1_Fines;
CREATE TABLE Project1_Fines(
  Loan_ID     VARCHAR2(40) NOT NULL,  --Pull from Loan_ID in
  Fine_Amt    NUMBER (4,2) NOT NULL,  --"Fine amount is $0.25 per day late"
  PAID        VARCHAR2(6)  NOT NULL,  --Indicate if the Fine is paid (No specification in project
guidelines)

  CONSTRAINT FinesPK PRIMARY KEY (Loan_ID),
  CONSTRAINT FinesFK FOREIGN KEY (Loan_ID)
      REFERENCES Project1_Book_Loans(Loan_ID)
  );

DELETE Project1_FINES;
INSERT INTO Project1_FINES
  SELECT  LOAN_ID,
        (Date_IN - Due_Date)*.25,
```

```
        'PAID'
    FROM Project1_Book_Loans WHERE Due_Date<Date_in;  --!!!! Will need to verify this once
Book_Loans work and can randomly assign Paid or Not paid
--SELECT * from Project1_FINES;
--Now adjust half of the fines to show as unpaid
UPDATE Project1_Fines SET Paid = 'Unpaid' WHERE rownum<=25;
--SELECT * from Project1_FINES;
```

BOOK SEARCH AND AVAILABILITY

I Created a Table Called Search Term To Easily Update The Search Query:

```
CREATE TABLE Search_Term(
    Term            VARCHAR2(50)    NOT NULL,   --Term designated by user
    Input_Branch    NUMBER (2)      NULL,       --Branch location
```

```
CONSTRAINT SearchPK PRIMARY KEY (Term)
);
```

I Could Then Update The Search Table Using The Following:

```
DELETE FROM Search_Term;
INSERT INTO Search_Term( Term, Input_Branch)
    VALUES ('Armand', NULL);          --'Armand',  'Turtle', '451','0395878063','Art Of the '
```

And Here Is The Search SQL With Single Search Functionality And Optional Branch Specification:

```
--Perform Search On All Books AND Adjust for Checked Out Books
SELECT ISBN, Title, Author, Sum(Available) AS Copies_Available FROM
        --THIS SECTION PULLS ALL BOOKS THAT FIT THE QUERY BUT DOESN'T ACCOUNT FOR BOOKS
CHECKED OUT
        (SELECT Book_ID, ISBN, Title, Author, Branch_ID, 1 AS Available
            FROM
                (SELECT Book_ID, ISBN AS ISBNExist, Branch_ID FROM Project1_Book_Copies
                    WHERE Branch_Id = (SELECT Input_Branch FROM Search_Term)
                        OR (Branch_ID IS NOT NULL AND (SELECT Input_Branch FROM Search_Term) IS
NULL)
                        )
            INNER JOIN
                (SELECT  BK.ISBN AS ISBN,
                        BK.Title AS Title,
                        LISTAGG(Auth.Name,', ') WITHIN GROUP (ORDER BY BK.ISBN) AS Author
                    FROM Project1_BOOKS BK
                    INNER JOIN  Project1_Book_Authors BKAuth ON BkAuth.ISBN = BK.ISBN
                    INNER JOIN  Project1_Authors Auth ON Auth.Author_ID = BKAuth.Author_ID
                    WHERE   BK.ISBN IN
                                (    SELECT ISBN AS SearchISBN From Project1_Books WHERE LOWER(title)
```

LIKE LOWER('%'||(Select Term From Search_Term)||'%') --Search In Book Titles
UNION
SELECT ISBN AS SearchISBN FROM Project1_Book_Authors
WHERE Author_ID IN
(SELECT Author_ID FROM Project1_Authors WHERE
Lower(Name) LIKE LOWER('%'||(Select Term From Search_Term)||'%')) --Search in Author Names
UNION
SELECT ISBN AS SearchISBN FROM Project1_Books WHERE LOWER(ISBN)
LIKE LOWER((Select Term From Search_Term)) --Search in ISBN
)
GROUP BY BK.ISBN,Bk.Title
ORDER BY BK.Title
)
ON ISBN = ISBNEXIST

```
    UNION
    --THIS SECTION PULLS ALL BOOKS THAT FIT THE QUERY AND ARE CURRENTLY CHECKED
OUT
        SELECT Bk_ID AS Book_ID, ISBN, Title, Author, Branch_ID, -1 AS Available

            FROM
                (SELECT Book_ID AS Bk_ID, ISBN AS ISBNExist, Branch_ID FROM
Project1_Book_Copies
                    WHERE Branch_Id = (SELECT Input_Branch FROM Search_Term)
                        OR (Branch_ID IS NOT NULL AND (SELECT Input_Branch FROM
Search_Term) IS NULL)
                        )
            INNER JOIN
                (SELECT  BK.ISBN AS ISBN,
                        BK.Title AS Title,
                        LISTAGG(Auth.Name,', ') WITHIN GROUP (ORDER BY BK.ISBN)
AS Author
                    FROM Project1_BOOKS BK
                    INNER JOIN  Project1_Book_Authors BKAuth ON BkAuth.ISBN =
BK.ISBN
                    INNER JOIN  Project1_Authors Auth ON Auth.Author_ID =
BKAuth.Author_ID
                    WHERE   BK.ISBN IN
                                        (    SELECT ISBN AS SearchISBN From
Project1_Books WHERE LOWER(title) LIKE LOWER('%'||(Select Term From
Search_Term)||'%') --Search In Book Titles
                                         UNION
                                             SELECT ISBN AS SearchISBN FROM
Project1_Book_Authors
                                                 WHERE Author_ID IN
                                                             (SELECT Author_ID
FROM Project1_Authors WHERE Lower(Name) LIKE LOWER('%'||(Select Term From
Search_Term)||'%')) --Search in Author Names
                                         UNION
                                             SELECT ISBN AS SearchISBN FROM
Project1_Books WHERE LOWER(ISBN) LIKE LOWER((Select Term From Search_Term)) --
Search in ISBN
                                        )
                    GROUP BY BK.ISBN,Bk.Title
                    ORDER BY BK.Title
```

```
                    )
                ON ISBN = ISBNEXIST
            Inner Join Project1_Book_Loans BL ON Bk_ID = BL.Book_ID
                WHERE Date_IN IS NULL
)
Group BY ISBN, Title,Author
```

;

EXAMPLE 1 Search ('live b', 1)

I choose this example because Book_ID 1006093315101OF1 is still checked out. This is a copy of ISBN 0060933151 at branch 1 Titled Everville written by Clive Barker

DELETE FROM Search_Term;
INSERT INTO Search_Term(Term, Input_Branch)
  VALUES ('live b', 1);

SEARCH RESULTS:

	ISBN	TITLE	AUTHOR	COPIES_AVAILABLE
1	0060182970	Coldheart Canyon: A Hollywood Ghost Story	Clive Barker	1
2	0061093084	Everville: The Second Book Of The Art	Clive Barker	1
3	0060933151	Everville	Clive Barker	0
4	0061092002	Galilee	Clive Barker	1
5	0451455126	Night Screams (Stalkers)	Clive Barker, David Morrell, Lawrence Block, Ray Brad	1
6	0060177241	The Thief Of Always: A Fable	Clive Barker	1
7	0805017224	The Complete Phantom Of The Opera (Owl Books)	Clive Barda, George Perry	1
8	0060924675	How To Live Between Office Visits: A Guide To Life, Love And Health	Bernie S. Siegel	1
9	006093316X	The Great And Secret Show	Clive Barker	1
10	006103018X	Coldheart Canyon	Clive Barker	1
11	0061053716	Imajica	Clive Barker	1
12	0061099015	The Great And Secret Show	Clive Barker	1
13	0061099643	Imajica	Clive Barker	1
14	0061091464	The Thief Of Always	Clive Barker	1
15	0930289595	The Sandman Vol. 2: The Doll's House	K. C. (editor) - Intro By Clive Bar, Neil; Carlson	1

You can see that row 8 returns a title with 'live b' string and the rest all have author Clive Barker. The copies available for 0060933151 also correctly shows a value of 0 as our checked out copy is the only copy at this branch.

EVIDENCE OF FUNCTIONALITY

	LOAN_ID	BOOK_ID	CARD_NO	DATE_OUT	DUE_DATE	DATE_IN
1	ID0000081006093315101OF115-JUL-21	1006093315101OF1	ID000008	17-OCT-21	17-NOV-21	(null)
2	ID0000201034535524501OF105-JUL-21	1034535524501OF1	ID000020	22-OCT-21	22-NOV-21	(null)
3	ID0000281034538657401OF225-JUL-21	1034538657401OF2	ID000028	25-OCT-21	25-NOV-21	(null)
4	ID0000471034543094801OF119-JUL-21	1034543094801OF1	ID000047	20-OCT-21	20-NOV-21	(null)
5	ID0000511037320142701OF108-JUL-21	1037320142701OF1	ID000051	14-OCT-21	14-NOV-21	(null)
6	ID0000581037571935001OF102-JUL-21	1037571935001OF1	ID000058	16-OCT-21	16-NOV-21	(null)
7	ID0000601039330158301OF117-JUL-21	1039330158301OF1	ID000060	25-OCT-21	25-NOV-21	(null)
8	ID0000621046500559401OF124-JUL-21	1046500559401OF1	ID000062	19-OCT-21	19-NOV-21	(null)
9	ID0000661055328524601OF126-JUL-21	1055328524601OF1	ID000066	15-OCT-21	15-NOV-21	(null)
10	ID0000711055358550901OF114-JUL-21	1055358550901OF1	ID000071	24-OCT-21	24-NOV-21	(null)

	BOOK_ID	ISBN	BRANCH_ID
1	1006093315101OF1	0060933151	1

	AUTHOR_ID	ISBN
1	00609331511	0060933151

	AUTHOR_ID	NAME
1	00609331511	Clive Barker

```
448 SELECT * FROM Project1_Book_Copies
449     WHERE ISBN = '0060933151' AND Branch_ID = 1;
```

Script Output × Query Result ×

SQL | All Rows Fetched: 1 in 0.052 seconds

	BOOK_ID	ISBN	BRANCH_ID
1	1006093315101OF1	0060933151	1

####

EXAMPLE 2 Search ('0060933151',NULL)

This is repurposing the same book but this time looking for it's ISBN without specifying a branch.

```
DELETE FROM Search_Term;
INSERT INTO Search_Term( Term, Input_Branch)
   VALUES ('0060933151', NULL);
```

SEARCH RESULTS:

	ISBN	TITLE	AUTHOR	COPIES_AVAILABLE
1	0060933151	Everville	Clive Barker	4

You can see 4 copies are available. Since we already know that 1 copy at branch 1 is checked out we expect to see 5 copies show up when we search Book_Copies for this ISBN which is verified below.

```
448 SELECT * FROM Project1_Book_Copies
449     WHERE ISBN = '0060933151' ;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.045 seconds

	BOOK_ID	ISBN	BRANCH_ID
1	10060933151010F1	0060933151	1
2	20060933151010F1	0060933151	2
3	30060933151010F1	0060933151	3
4	40060933151010F1	0060933151	4
5	50060933151010F1	0060933151	5

EXAMPLE 3 Search ('451', NULL)

I chose '451' to show that ISBN search will not show variations of '%451%' in the search results.

```
DELETE FROM Search_Term;
INSERT INTO Search_Term( Term, Input_Branch)
   VALUES ('451', NULL);
```

SEARCH RESULTS:

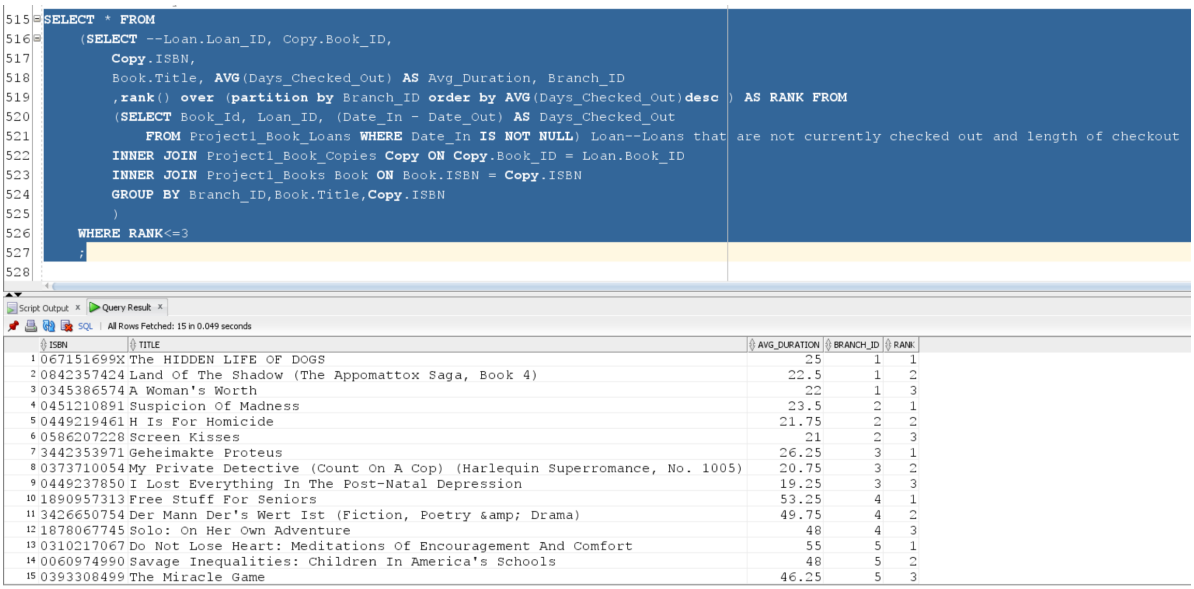
	ISBN	TITLE	AUTHOR	COPIES_AVAILABLE
1	0395878063	Fahrenheit 451: And Related Readings (Literature Connections)	Ray Bradbury	4
2	067187036X	Fahrenheit 451: A Novel	Ray Bradbury	5
3	9505470010	Fahrenheit 451 (Spanish Edition)	Ray Bradbury	4
4	0345342968	Fahrenheit 451	Ray Bradbury	4
5	3257208626	Fahrenheit 451	Ray Bradbury	2
6	3453164121	Fahrenheit 451	Brian W. Aldiss, Ray Bradbury	2
7	0345410017	Fahrenheit 451	Ray Bradbury	5
8	0345292340	Fahrenheit 451	Ray Bradbury	4

CUSTOM REPORTS

REPORT A

```
--REPORT A  The purchasing department is interested in seeing if there are reasons
--        as to why some  books stay out for longer periods of time.  They want
--        to start there investigation by receiving a report for the following:
--Display the 3 highest Average Checkout Durations per ISBN/Book Title By Branch.
--        Only consider books that have been checked back in

SELECT * FROM
   (SELECT --Loan.Loan_ID, Copy.Book_ID,
      Copy.ISBN,
      Book.Title, AVG(Days_Checked_Out) AS Avg_Duration, Branch_ID
      ,rank() over (partition by Branch_ID order by AVG(Days_Checked_Out)desc ) AS RANK FROM
      (SELECT Book_Id, Loan_ID, (Date_In - Date_Out) AS Days_Checked_Out
         FROM Project1_Book_Loans WHERE Date_In IS NOT NULL) Loan--Loans that are not
currently checked out and length of checkout
      INNER JOIN Project1_Book_Copies Copy ON Copy.Book_ID = Loan.Book_ID
      INNER JOIN Project1_Books Book ON Book.ISBN = Copy.ISBN
      GROUP BY Branch_ID,Book.Title,Copy.ISBN
      )
   WHERE RANK<=3
   ;
```



Script Output | Query Result

All Rows Fetched: 15 in 0.049 seconds

	ISBN	TITLE	AVG_DURATION	BRANCH_ID	RANK
1	067151699X	The HIDDEN LIFE OF DOGS	25	1	1
2	0842357424	Land Of The Shadow (The Appomattox Saga, Book 4)	22.5	1	2
3	0345386574	A Woman's Worth	22	1	3
4	0451210891	Suspicion Of Madness	23.5	2	1
5	0449219461	H Is For Homicide	21.75	2	2
6	0586207228	Screen Kisses	21	2	3
7	3442353971	Geheimakte Proteus	26.25	3	1
8	0373710054	My Private Detective (Count On A Cop) (Harlequin Superromance, No. 1005)	20.75	3	2
9	0449237850	I Lost Everything In The Post-Natal Depression	19.25	3	3
10	1890957313	Free Stuff For Seniors	53.25	4	1
11	3426650754	Der Mann Der's Wert Ist (Fiction, Poetry & Drama)	49.75	4	2
12	1878067745	Solo: On Her Own Adventure	48	4	3
13	0310217067	Do Not Lose Heart: Meditations Of Encouragement And Comfort	55	5	1
14	0060974990	Savage Inequalities: Children In America's Schools	48	5	2
15	0393308499	The Miracle Game	46.25	5	3

REPORT B

```
--Report B Marketing team wants to gain insights into all borrowers who check
--     out books from multiple libraries.
--Display the following
--     First and Last Name, Card_No,
--     Total # of locations a borrower checked out at,
--     Total # of checkouts at all locations for the borrower,
--     Sum of # of Days the borrower's books were cheked out,
--     And the average # of days each book was checked out by the borrower
--Only display borrowers who checked out books at more than 1 location
--Only consider books that have been returned
--Display in ascending alphabetical order by last name then first name
```

```
SELECT First_Name, Last_Name, Card_No,
      Count(DISTINCT(Branch_ID)) AS Checkout_Locations,
      Count(Loan_ID) AS Number_Borrowed,
      SUM(Date_IN - Date_Out) AS Total_Days_Borrowed,
      AVG(Date_In - Date_Out) AS Average_Days_Borrowed
   FROM(
      SELECT Bor.FName As First_Name, Bor.LName AS Last_Name,
            Bor.Card_No, Loan.Book_Id, Loan.Loan_ID,
            Branch_ID, Loan.Date_Out, Loan.Date_In
      FROM
         Project1_Book_Loans Loan--Consider all Loans regardless of checked in status
         INNER JOIN Project1_Borrower Bor ON Loan.Card_No = Bor.Card_No
         INNER JOIN Project1_Book_Copies Copy ON Copy.Book_Id = Loan.Book_ID
      WHERE Loan.Date_In IS NOT NULL
   )
   GROUP BY Card_No,Last_Name, First_Name
      HAVING Count(DISTINCT(Branch_ID))>1
   ORDER BY Last_Name, First_Name
;
```

```
544 SELECT First_Name, Last_Name, Card_No,
545         Count(DISTINCT(Branch_ID)) AS Checkout_Locations,
546         Count(Loan_ID) AS Number_Borrowed,
547         SUM(Date_IN - Date_Out) AS Total_Days_Borrowed,
548         AVG(Date_In - Date_Out) AS Average_Days_Borrowed
549     FROM(
550         SELECT Bor.FName As First_Name, Bor.LName AS Last_Name,
551                 Bor.Card_No, Loan.Book_Id, Loan.Loan_ID,
552                 Branch_ID, Loan.Date_Out, Loan.Date_In
553         FROM
554             Project1_Book_Loans Loan--Consider all Loans regardless of checked in status
555             INNER JOIN Project1_Borrower Bor ON Loan.Card_No = Bor.Card_No
556             INNER JOIN Project1_Book_Copies Copy ON Copy.Book_Id = Loan.Book_ID
557         WHERE Loan.Date_In IS NOT NULL
558     )
559     GROUP BY Card_No,Last_Name, First_Name
560         HAVING Count(DISTINCT(Branch_ID))>1
561     ORDER BY Last_Name, First_Name
562 ;
563
```

Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.064 seconds

	FIRST_NAME	LAST_NAME	CARD_NO	CHECKOUT_LOCATIONS	NUMBER_BORROWED	TOTAL_DAYS_BORROWED	AVERAGE_DAYS_BORROWED
1	Antonio	Bishop	ID000123	2	2	32	16
2	Doris	Carpenter	ID000825	2	2	29	14.5
3	Aaron	Ellis	ID000299	2	2	48	24
4	Catherine	Harris	ID000186	2	2	22	11
5	Edward	Harvey	ID000694	2	2	13	6.5
6	Marilyn	Jordan	ID000499	2	2	37	18.5
7	Phyllis	Long	ID000630	2	2	16	8
8	Gregory	Marshall	ID000406	2	2	88	44
9	Jeremy	Rogers	ID000910	2	2	105	52.5

REPORT C

```
--Report C  The Collections Department would like to audit all past and current
--          borrowers who have generted fines.
--Provide   Borrower Card_No, Borrower First and Last Name, Borrower Email,
--          Total # of Fines, Total Fine Amount, Current Outstanding Fines
```

--Display Data in Descending order of outstanding balance and then Total Fine Amount

SELECT BCardNo AS Card_No, FName AS First_Name, LName AS Last_Name, Email, Number_Of_Fines, Total_Fine_Amt, Outstanding_Balance
  FROM(
    SELECT CrdNo AS BCardNo, Count(Loan_ID) AS Number_Of_Fines, SUM(Paid_Fine + Unpaid_Fine) AS Total_Fine_Amt, SUM(Unpaid_Fine) AS Outstanding_Balance
      FROM(
        SELECT Loan.Loan_ID, Loan.Card_No AS CrdNo, Fine_Amt AS Paid_Fine, 0 AS UNPAID_FINE
          FROM Project1_Book_Loans Loan
          INNER JOIN Project1_Fines Fine ON Loan.Loan_ID = Fine.Loan_ID
          WHERE Paid = 'PAID'
        UNION
        SELECT Loan.Loan_ID, Card_No AS CrdNo, 0 AS PAID_FINE, Fine_Amt AS UNPAID_Fine
          FROM Project1_Book_Loans Loan
          INNER JOIN Project1_Fines Fine ON Loan.Loan_ID = Fine.Loan_ID
          WHERE Paid = 'Unpaid'
)

```
        GROUP BY CrdNo
    )
  INNER JOIN Project1_Borrower Bor ON BCardNo = Bor.Card_No
  ORDER BY Outstanding_Balance DESC, Total_Fine_AMT DESC
;
```

```
SELECT BCardNo AS Card_No, FName AS First_Name, LName AS Last_Name, Email, Number_Of_Fines, Total_Fine_Amt, Outstanding_Balance
    FROM(
        SELECT CrdNo AS BCardNo, Count(Loan_ID) AS Number_Of_Fines, SUM(Paid_Fine + Unpaid_Fine) AS Total_Fine_Amt, SUM(Unpaid_Fine) AS Outstanding_Balance
            FROM(
                SELECT Loan.Loan_ID, Loan.Card_No AS CrdNo, Fine_Amt AS Paid_Fine, 0 AS UNPAID_FINE
                    FROM Project1_Book_Loans Loan
                    INNER JOIN Project1_Fines Fine ON Loan.Loan_ID = Fine.Loan_ID
                    WHERE Paid = 'PAID'
                UNION
                SELECT Loan.Loan_ID, Card_No AS CrdNo, 0 AS PAID_FINE, Fine_Amt AS UNPAID_Fine
                    FROM Project1_Book_Loans Loan
                    INNER JOIN Project1_Fines Fine ON Loan.Loan_ID = Fine.Loan_ID
                    WHERE Paid = 'Unpaid'
            )

            GROUP BY CrdNo
        )
    INNER JOIN Project1_Borrower Bor ON BCardNo = Bor.Card_No
    ORDER BY Outstanding_Balance DESC, Total_Fine_AMT DESC
    ;
```

Script Output | Query Result

All Rows Fetched: 27 in 0.045 seconds

	CARD_NO	FIRST_NAME	LAST_NAME	EMAIL	NUMBER_OF_FINES	TOTAL_FINE_AMT	OUTSTANDING_BALANCE
1	ID000368	Ruby	Anderson	randersona7@slashdot.org	2	11.5	11.5
2	ID000886	Carolyn	Gordon	cgordonol@icq.com	2	11.25	11.25
3	ID000405	Doris	Peterson	dpetersonb8@engadget.com	2	10.25	10.25
4	ID000903	Lillian	Stone	lstonep2@goo.ne.jp	2	9.75	9.75
5	ID000403	Nicole	Bell	nbellb6@delicious.com	2	9	9
6	ID000370	Melissa	Perkins	mperkinsa9@psu.edu	2	8.5	8.5
7	ID000385	Louis	Elliott	lelliottao@mit.edu	2	8.25	8.25
8	ID000907	Tammy	Robinson	trobinsonp6@economist.com	2	6.5	6.5
9	ID000369	Steven	Bailey	sbaileya8@mlb.com	2	6	6
10	ID000909	Earl	White	ewhitep8@slashdot.org	2	5.5	5.25
11	ID000869	Scott	Chavez	schavezo4@php.net	1	4.25	4.25
12	ID000884	Tina	Rogers	trogersoj@mysql.com	2	4	4
13	ID000906	Lori	Hunter	lhunterp5@about.com	2	1.75	1.75
14	ID000366	Roy	Turner	rturnera5@abc.net.au	1	0.5	0.5
15	ID000418	Wayne	Franklin	wfranklinbl@instagram.com	2	13.25	0
16	ID000910	Jeremy	Rogers	jrogersp9@squarespace.com	2	10.75	0
17	ID000911	Earl	Chapman	echapmanpa@telegraph.co.uk	2	9.75	0
18	ID000411	Dennis	Duncan	dduncanbe@marriott.com	2	9.25	0
19	ID000921	Linda	Garcia	lgarciapk@imageshack.us	2	8	0
20	ID000430	Karen	Sanchez	ksanchezbx@census.gov	2	7.5	0
21	ID000913	Anna	Lopez	alopezpc@google.pl	2	7.5	0
22	ID000438	Janice	Garrett	jgarrettc5@i2i.jp	2	7.25	0
23	ID000406	Gregory	Marshall	gmarshallb9@narod.ru	2	6.5	0
24	ID000925	Deborah	Medina	dmedinapo@sfgate.com	1	6	0
25	ID000915	Angela	Nguyen	anguyenpe@163.com	2	3.5	0
26	ID000421	Alan	Lynch	alynchbo@dailymail.co.uk	2	2.25	0
27	ID000441	Laura	Chapman	lchapmanc8@google.it	1	0.5	0