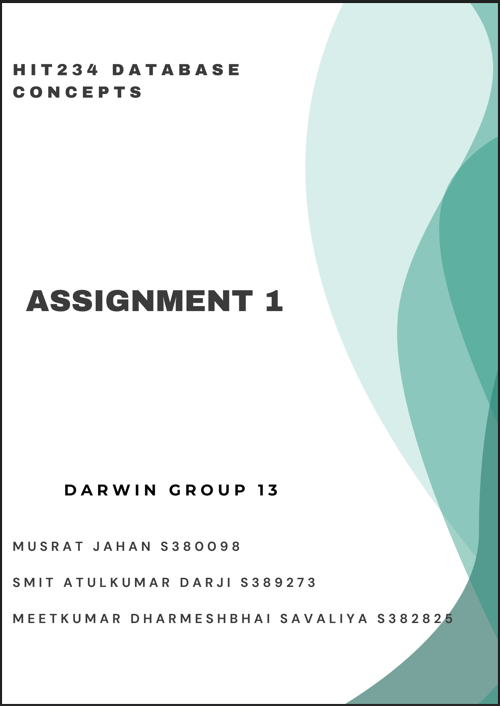
****

**Part A - Single and multiple table queries**

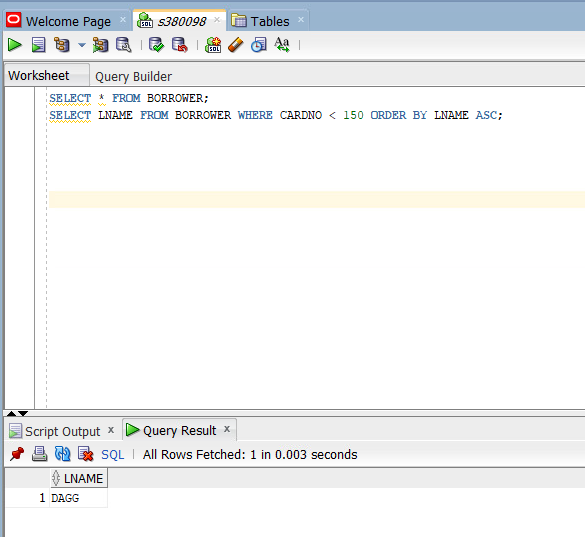
**Questions 1 – 5:**

1. List in ascending order the Last Names of all borrowers with Card Number less than 150.

**Answer:**

SELECT LNAME FROM BORROWER WHERE CARDNO < 150 ORDER BY LNAME ASC;

**Output:**

****

1. What are the records of those loans between 3 June 2014 and 8 Mar 2018?
   * The output should include the name of the cardholder, and title of the book
   * List in ascending order by last name and first name

**Answer:**

SELECT B.LNAME, B.FNAME, BK.TITLE FROM LOAN L JOIN BORROWER B ON L.CARDNO = B.CARDNO JOIN BOOK BK ON L.ISBN = BK.ISBN WHERE L.DATEOUT BETWEEN TO\_DATE (’03-JUN-2014’, ‘DD-MM-YYYY’) AND TO\_DATE(’08-MAR-2018’, ‘DD-MON-YYYY’) ORDER BY B.LNAME ASC , B.FNAME ASC;

**Output:**



1. Find the total numbers of loans with last name beginning with D through E (including E); include last name beginning with Q as well as Z also.
   * The output of the above question should be a column containing the sum of all the loan based on the condition in the query, alternatively the output could be a table with columns that has the last name as per the condition in the query condition, also the table contains the list of loans.

**Answer:**

SELECT B.LNAME, COUNT(\*) AS total\_numbers\_Loans

FROM LOAN L

JOIN BORROWER B ON L.CARDNO = B.CARDNO

WHERE (SUBSTR(B.LNAME, 1, 1) BETWEEN 'D' AND 'E')

OR SUBSTR(B.LNAME, 1, 1) IN ('Q', 'Z')

GROUP BY B.LNAME

ORDER BY B.LNAME;

SELECT B.LNAME, COUNT(\*) AS total\_numbers\_Loans

FROM LOAN L

JOIN BORROWER B ON L.CARDNO = B.CARDNO

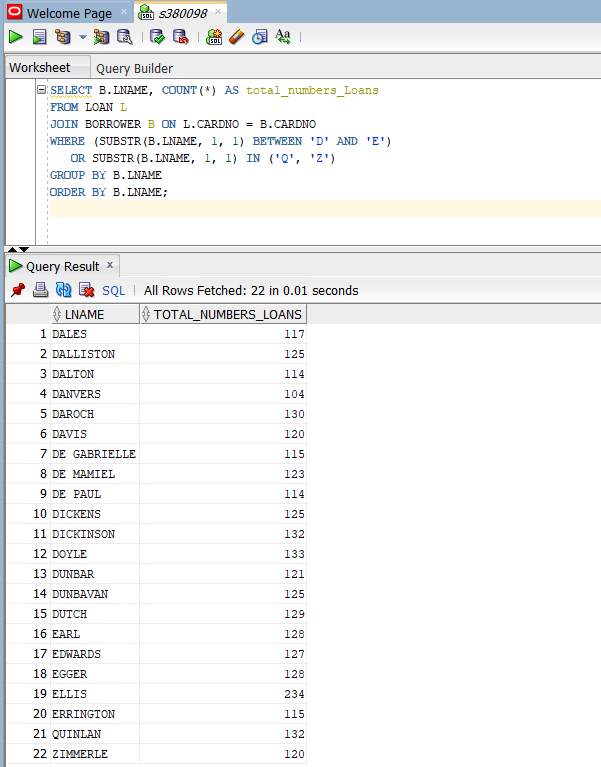
WHERE (SUBSTR(B.LNAME, 1, 1) BETWEEN 'D' AND 'E')

OR SUBSTR(B.LNAME, 1, 1) IN ('Q', 'Z')

GROUP BY B.LNAME,

ORDER BY B.LNAME;

**Output:**



1. Find all borrowers for a loan that have the Date-in before 15-March-15 and the Card number between 100 and 300 in Karama or Darwin.
   * The output for the above question is a list of all borrowers (first name and last name) based on the condition in the query.

**Answer:**

SELECT DISTINCT B.FNAME, B.LNAME

FROM LOAN L

JOIN BORROWER B ON L.CARDNO = B.CARDNO

WHERE L.DATEIN IS NOT NULL

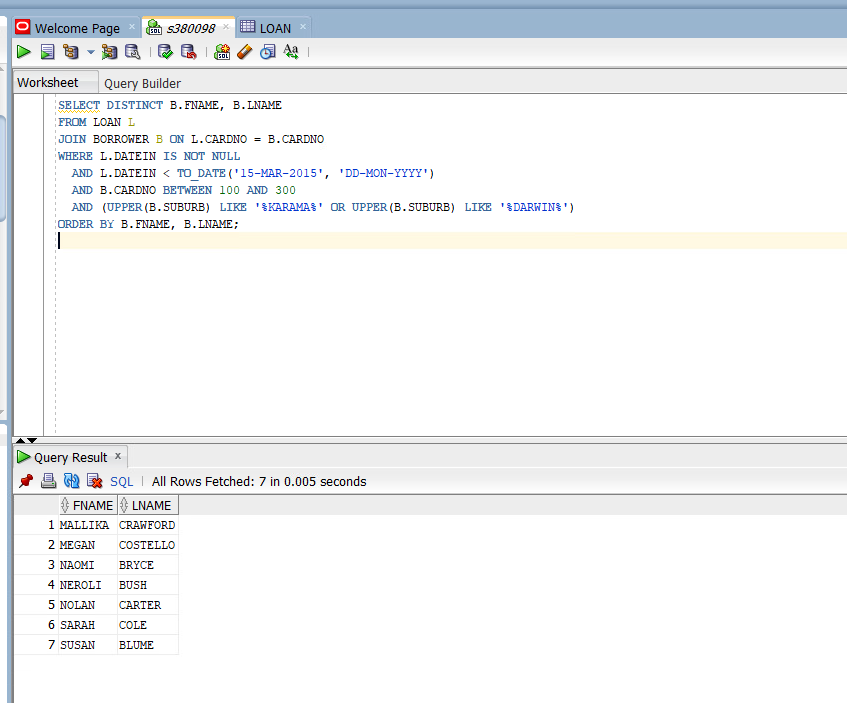
AND L.DATEIN < TO\_DATE('15-MAR-2015', 'DD-MON-YYYY')

AND B.CARDNO BETWEEN 100 AND 300

AND (UPPER(B.SUBURB) LIKE '%KARAMA%' OR UPPER(B.SUBURB) LIKE '%DARWIN%')

ORDER BY B.FNAME, B.LNAME;

**Output:**



1. Find the number of loans, which have been made from each branch?
   * Note all branches need to be included even those with no loans
   * The output of the above question is a table containing the list of loans only as per the condition in the query

**Answer:**

SELECT DISTINCT B.FNAME, B.LNAME

FROM LOAN L

JOIN BORROWER B ON L.CARDNO = B.CARDNO

WHERE L.DATEIN IS NOT NULL

AND L.DATEIN < TO\_DATE('15-MAR-2015', 'DD-MON-YYYY')

AND B.CARDNO BETWEEN 100 AND 300

AND (UPPER(B.SUBURB) LIKE '%KARAMA%' OR UPPER(B.SUBURB) LIKE '%DARWIN%')

ORDER BY B.FNAME, B.LNAME;

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**Question 6:**

1. Create your own question based on the library database, and also provide a SQL Statement to answer your question.

**Answer:**

Find the names of all borrowers who have borrowed more than 3 books, along with the total number of books they have borrowed.

SELECT B.FNAME, B.LNAME, COUNT(L.ISBN) AS NUM\_BOOKS\_BORROWED FROM BORROWER B

JOIN LOAN L ON B.CARDNO = L.CARDNO

JOIN BOOK BK ON L.ISBN = BK.ISBN

GROUP BY B.FNAME, B.LNAME

HAVING COUNT(L.ISBN) > 3

ORDER BY NUM\_BOOKS\_BORROWED ASC;

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**Part B – Table and Advanced Queries**

1. The following Gym application database for weight session recording is as follows. Write and run SQL statements for creating the Occurrence\_Exercise table.
   1. Include constraint values for the Condition Column: Heavy, Medium, Light

**Answer:**

CREATE TABLE Exercise

(EXERCISENO INT PRIMARY KEY,

EXERCISEDESCRIPTION VARCHAR2(255),

EXERCISETYPE VARCHAR2(100)

);

CREATE TABLE Occurrence

(OCCURANCEID INT PRIMARY KEY,

OCCDATE DATE,

OCCPLACE VARCHAR2(255),

OCCSTARTTIME DATE,

OCCFINISHTIME DATE);

CREATE TABLE Occurrence\_Exercise

(OCCEXID INT PRIMARY KEY,

EXERCISENO INT,

OCCURANCEID INT,

REP INT,

ATTEMPT1 INT,

ATTEMPT2 INT,

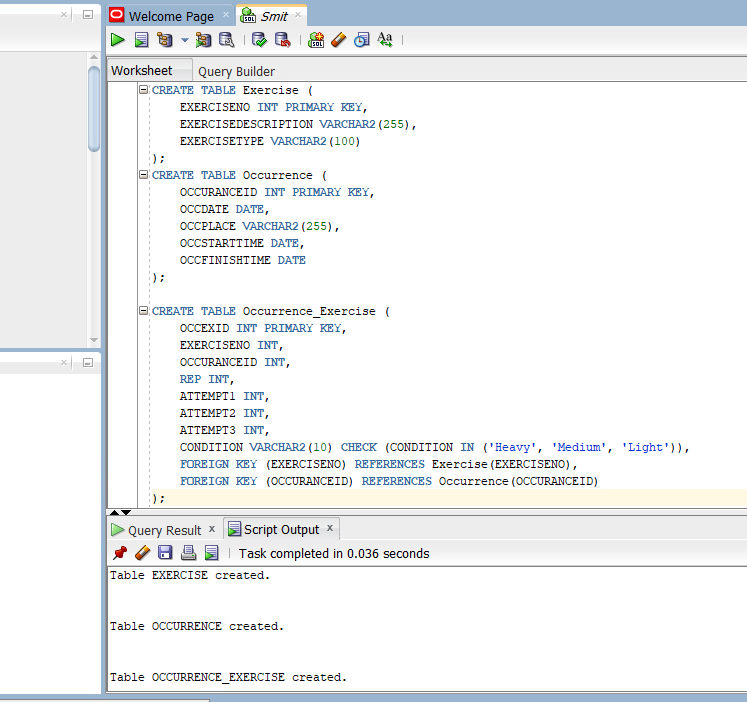
ATTEMPT3 INT,

CONDITION VARCHAR2(10) CHECK (CONDITION IN ('Heavy', 'Medium', 'Light')),

FOREIGN KEY (EXERCISENO) REFERENCES Exercise (EXERCISENO), FOREIGN KEY (OCCURANCEID) REFERENCES Occurrence (OCCURANCEID)

);

**Output:**



1. Write an SQL statement that will insert the data into **Occurrence\_Exercise** table only.
   1. Definition of rep stands for repetition.
   2. Each exercise attempt has the rep number as indicated in the occurrence table.

**Answer:**

INSERT INTO Occurrence\_Exercise (OCCEXID, EXERCISENO, OCCURRENCEID, REP, ATTEMPT1, ATTEMPT2, ATTEMPT3, CONDITION)

VALUES (9001, 1, 1, 8, 15.00, 15.00, 15.00, 'Light');

INSERT INTO Occurrence\_Exercise (OCCEXID, EXERCISENO, OCCURRENCEID, REP, ATTEMPT1, ATTEMPT2, ATTEMPT3, CONDITION)

VALUES (9002, 2, 1, 8, 70.00, 75.00, 70.00, 'Medium');

INSERT INTO Occurrence\_Exercise (OCCEXID, EXERCISENO, OCCURRENCEID, REP, ATTEMPT1, ATTEMPT2, ATTEMPT3, CONDITION)

VALUES (9003, 3, 2, 12, 90.00, 95.00, 90.00, 'Heavy');

INSERT INTO Occurrence\_Exercise (OCCEXID, EXERCISENO, OCCURRENCEID, REP, ATTEMPT1, ATTEMPT2, ATTEMPT3, CONDITION)

VALUES (9004, 4, 3, 12, 200.00, 200.00, 200.00, 'Medium');

INSERT INTO Occurrence\_Exercise (OCCEXID, EXERCISENO, OCCURRENCEID, REP, ATTEMPT1, ATTEMPT2, ATTEMPT3, CONDITION)

VALUES (9005, 6, 3, 8, 110.00, 110.00, 115.00, 'Heavy');

INSERT INTO Occurrence\_Exercise (OCCEXID, EXERCISENO, OCCURRENCEID, REP, ATTEMPT1, ATTEMPT2, ATTEMPT3, CONDITION)

VALUES (9006, 5, 4, 10, 20.00, 25.00, NULL, 'Medium');

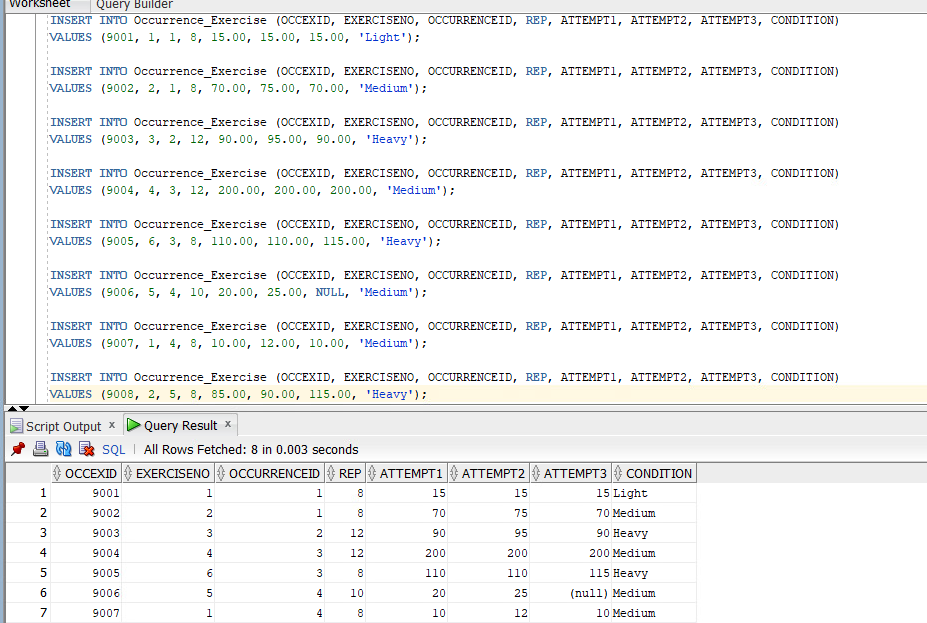
INSERT INTO Occurrence\_Exercise (OCCEXID, EXERCISENO, OCCURRENCEID, REP, ATTEMPT1, ATTEMPT2, ATTEMPT3, CONDITION)

VALUES (9007, 1, 4, 8, 10.00, 12.00, 10.00, 'Medium');

INSERT INTO Occurrence\_Exercise (OCCEXID, EXERCISENO, OCCURRENCEID, REP, ATTEMPT1, ATTEMPT2, ATTEMPT3, CONDITION)

VALUES (9008, 2, 5, 8, 85.00, 90.00, 115.00, 'Heavy');

**Output:**



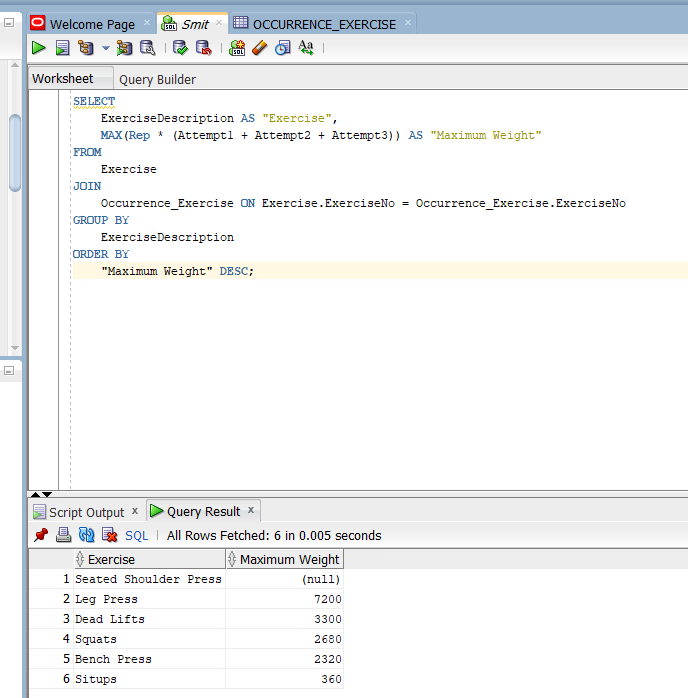
1. Find the exercise which contains the maximum total weight lifted
   * Show: Exercise, Maximum weight
   * Ensure you show captions for each of the columns
   * Hint: you would need to consider the number of times the weight is lifted and weight

e.g. rep multiplied by attempt multiplied by the weight

**Answer:**

SELECT ExerciseDescription AS "Exercise", MAX(Rep \* (Attempt1 + Attempt2 + Attempt3)) AS "Maximum Weight" FROM Exercise JOIN Occurrence\_Exercise ON Exercise.ExerciseNo = Occurrence\_Exercise.ExerciseNo GROUP BY ExerciseDescription ORDER BY "Maximum Weight" DESC;

**Output:**

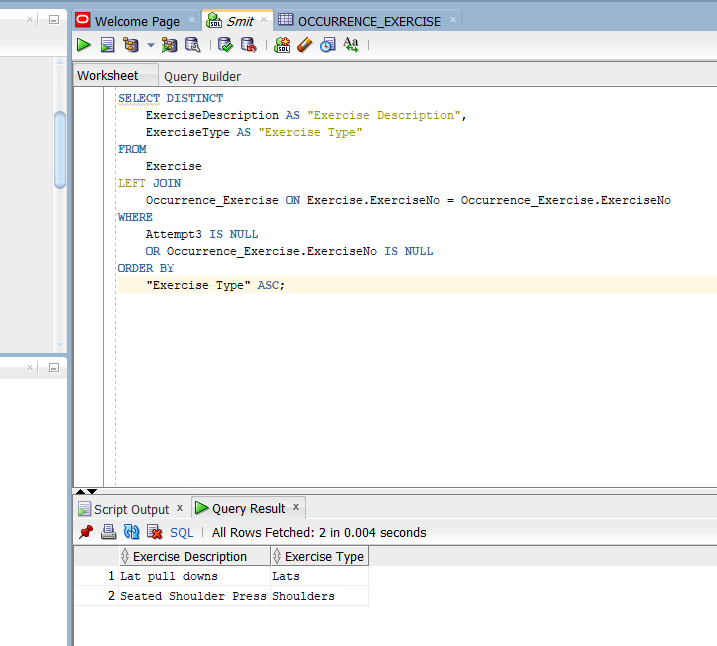


1. Find the exercise that the user was not able to complete 3 attempts in, or the exercise that the user has not attempted.
   1. Show appropriate columns
   2. List in ascending order exercise type
   3. Do not show repeated names
   4. Ensure you show captions for each of the columns

**Answer:**

SELECT DISTINCT ExerciseDescription AS "Exercise Description", ExerciseType AS "Exercise Type" FROM Exercise LEFT JOIN Occurrence\_Exercise ON Exercise.ExerciseNo = Occurrence\_Exercise.ExerciseNo WHERE Attempt3 IS NULL OR Occurrence\_Exercise.ExerciseNo IS NULL ORDER BY "Exercise Type" ASC;

**Output:**



**Part C – All Topics**

**Questions Using the Library Database**

1. What are the 5 most popular books?
   * Show ASBN, Title.
   * HINT: To limit results use ROWNUM.

**Answer:**

SELECT BR.BRANCHNAME, COUNT(L.BRANCHID) AS NUM\_LOANS

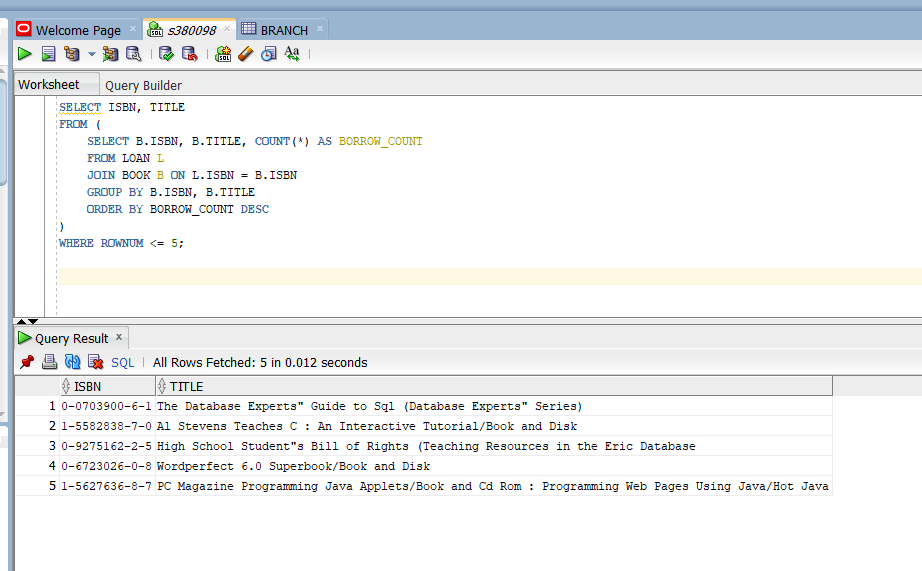
FROM BRANCH BR

LEFT JOIN LOAN L ON BR.BRANCHID = L.BRANCHID

GROUP BY BR.BRANCHNAME

ORDER BY BR.BRANCHNAME;

**Output:**



1. Which branches currently have no loans out?
   * Show all branch names.
   * List in ascending order by branch.

**Answer:**

SELECT ISBN, TITLE

FROM (

SELECT B.ISBN, B.TITLE, COUNT(\*) AS BORROW\_COUNT

FROM LOAN L

JOIN BOOK B ON L.ISBN = B.ISBN

GROUP BY B.ISBN, B.TITLE

ORDER BY BORROW\_COUNT DESC

)

WHERE ROWNUM <= 5;

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

1. Find the **longest time** a book has been loaned out for each branch
   * Show only branch name and number of days.
   * List in ascending order by branch name.
   * Show all branches, even if there are no loaned out books. Therefore, a branch may have a null value for the longest loaded out book.
   * Format the number of days to one decimal place.

**Answer:**

SELECT B.BRANCHNAME

FROM BRANCH B

LEFT JOIN LOAN L ON B.BRANCHID = L.BRANCHID

WHERE L.BRANCHID IS NULL

ORDER BY B.BRANCHNAME ASC;

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

1. Find the borrower that has returned a book on the dated 16 December 2010 and 8 May 2012
   * Show two different methods.

**Answer:**

**In method:**SELECT B.FNAME, B.LNAME

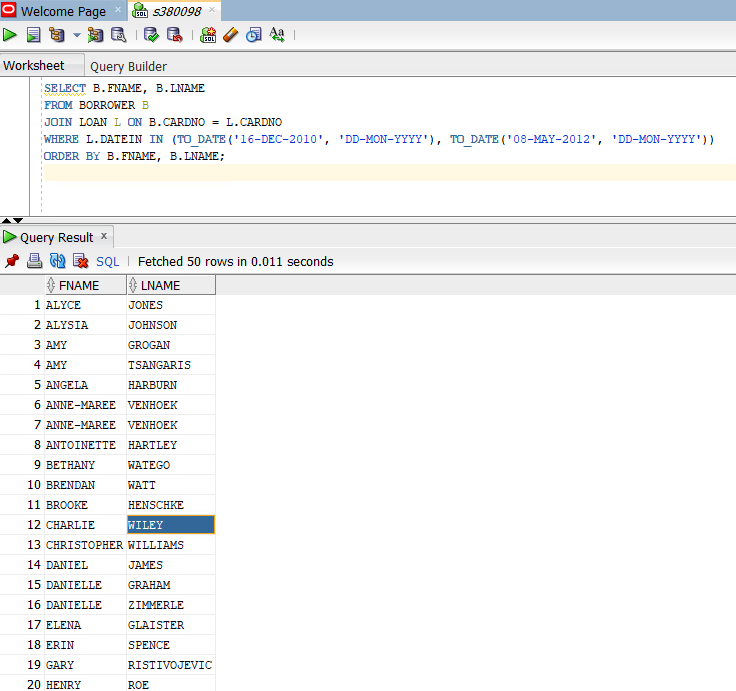
FROM BORROWER B

JOIN LOAN L ON B.CARDNO = L.CARDNO

WHERE L.DATEIN IN (TO\_DATE('16-DEC-2010', 'DD-MON-YYYY'), TO\_DATE('08-MAY-2012', 'DD-MON-YYYY'))

ORDER BY B.FNAME, B.LNAME;

**Output:**



**OR method:**

SELECT B.FNAME, B.LNAME

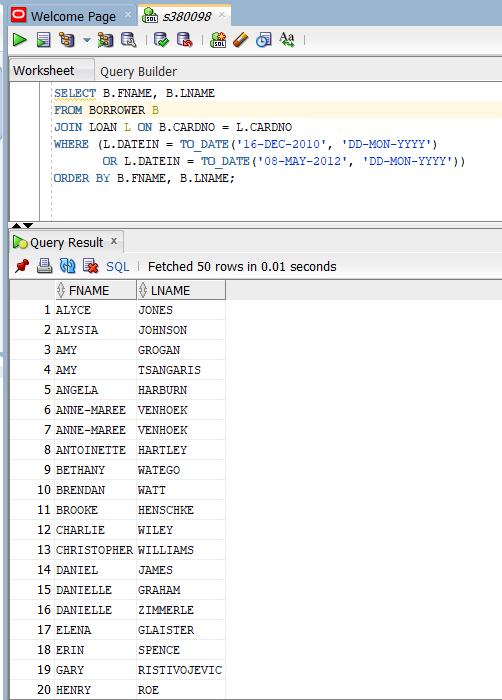
FROM BORROWER B

JOIN LOAN L ON B.CARDNO = L.CARDNO

WHERE (L.DATEIN = TO\_DATE('16-DEC-2010', 'DD-MON-YYYY')

OR L.DATEIN = TO\_DATE('08-MAY-2012', 'DD-MON-YYYY'))  
ORDER BY B.FNAME, B.LNAME;

**Output:**



**Question Using the CDU Customer Invoice Database**

1. Devise a delete query, which will eliminate duplicates for customer columns `ID`,

`CustomerLastName`, `CustomerFirstName` and invoice columns `Invoice\_Date` and

`Invoice\_No`.

* + Ensure the query removes all duplicate data.

**Answer:**

DELETE FROM CDUINVOICE

WHERE ROWID NOT IN (

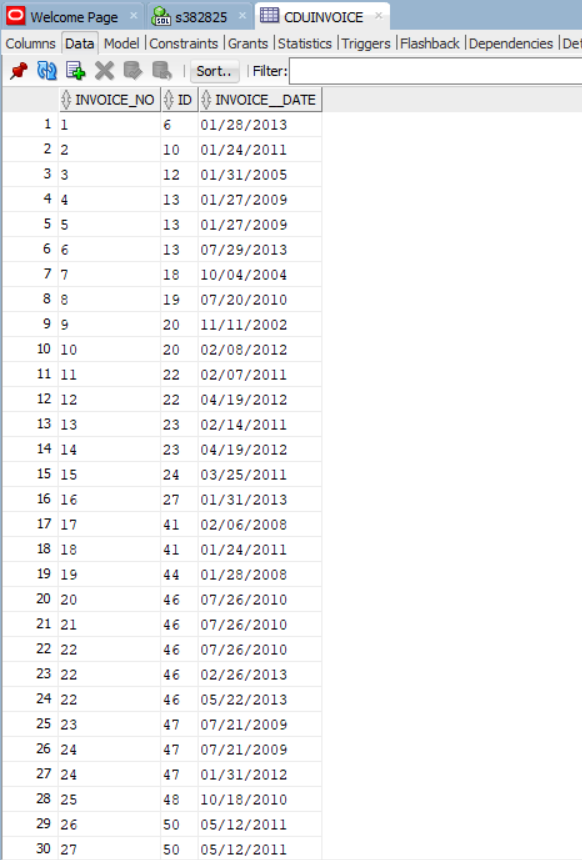
SELECT MIN(ROWID)

FROM CDUINVOICE

GROUP BY ID

);

**Output**:  
 Before execution:



After Execution:

