## Que.Create a Book and Member Table for Library Management System

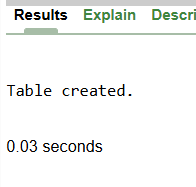
1. **Book Table:**

Create table Book1

(Book\_id number(10), book\_name varchar(20), author varchar(20),

publication\_year number(10));

## Output:



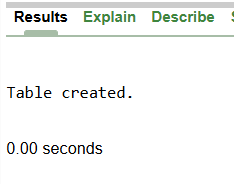
1. **Member Table**

Create table Member1

(Member\_id number(10), name varchar(20),

email varchar(20));

## Output:



**Que. Insert the Records into Book and Member Table. 1:Insert into Book table:**

INSERT INTO Book1 (Book\_id, book\_name, author, publication\_year) VALUES (1, 'The Great Gatsby', 'F. Scott Fitzgerald', 1925);

INSERT INTO Book1 (Book\_id, book\_name, author, publication\_year) VALUES (2, '1984', 'George Orwell', 1949);

INSERT INTO Book1 (Book\_id, book\_name, author, publication\_year) VALUES (3, 'To Kill a Mockingbird', 'Harper Lee', 1960);

INSERT INTO Book1 (Book\_id, book\_name, author, publication\_year) VALUES (4, 'Pride and Prejudice', 'Jane Austen', 1813);

INSERT INTO Book1 (Book\_id, book\_name, author, publication\_year) VALUES (5, 'The Catcher in the Rye', 'J.D. Salinger', 1951);

INSERT INTO Book1 (Book\_id, book\_name, author, publication\_year) VALUES (6, 'Moby Dick', 'Herman Melville', 1851);

INSERT INTO Book1 (Book\_id, book\_name, author, publication\_year) VALUES (7, 'War and Peace', 'Leo Tolstoy', 1869);

INSERT INTO Book1 (Book\_id, book\_name, author, publication\_year) VALUES (8, 'Brave New World', 'Aldous Huxley', 1932);

INSERT INTO Book1 (Book\_id, book\_name, author, publication\_year) VALUES (9, 'The Odyssey', 'Homer', 800);

INSERT INTO Book1 (Book\_id, book\_name, author, publication\_year) VALUES (10, 'The Picture of Dorian Gray', 'Oscar Wilde', 1890);

## Select Book table:

select \* from Book1

## 1:Insert into Member table:

INSERT INTO Member1 (Member\_id, name, email) VALUES (1, 'John Doe', 'john.doe@example.com');

INSERT INTO Member1 (Member\_id, name, email) VALUES (2, 'Jane Smith', 'jane.smith@example.com');

INSERT INTO Member1 (Member\_id, name, email) VALUES (3, 'Alice Johnson', 'alice.johnson@example.com');

INSERT INTO Member1 (Member\_id, name, email) VALUES (4, 'Bob Brown', 'bob.brown@example.com');

INSERT INTO Member1 (Member\_id, name, email) VALUES (5, 'Carol White', 'carol.white@example.com');

INSERT INTO Member1 (Member\_id, name, email) VALUES (6, 'David Wilson', 'david.wilson@example.com');

INSERT INTO Member1 (Member\_id, name, email) VALUES (7, 'Eva Green', 'eva.green@example.com');

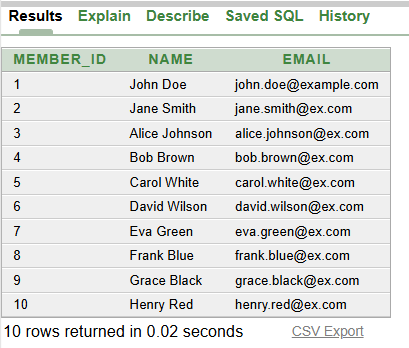
INSERT INTO Member1 (Member\_id, name, email) VALUES (8, 'Frank Blue', 'frank.blue@example.com');

INSERT INTO Member1 (Member\_id, name, email) VALUES (9, 'Grace Black', 'grace.black@example.com');

INSERT INTO Member1 (Member\_id, name, email) VALUES (10, 'Henry Red', 'henry.red@example.com');

## Select member table:

select \* from member

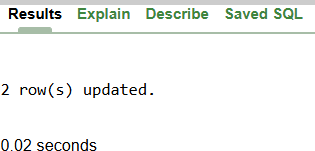


## Que.Perform Update Query in Book and Member Table.

1. **Update Query on Book table**

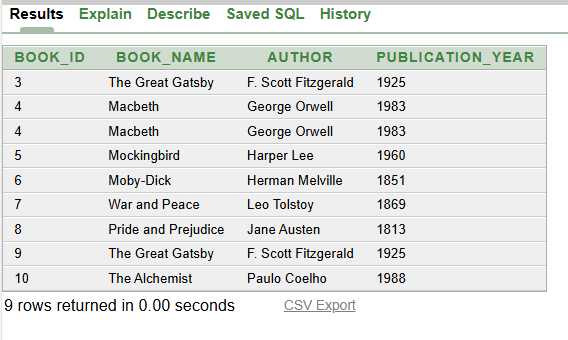
UPDATE Book1

SET book\_name = 'Macbeth', publication\_year = 1983 WHERE Book\_id = 4;



## Select Query:

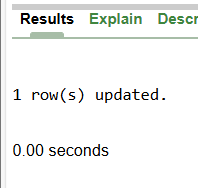
Select \* from Book1



## Update Query on Member table

UPDATE Member1

SET name = 'Sakshi', email = 'sakshi@gmail.com' WHERE Member\_id = 1;



## Que.Write SQL statement to make use of all phases of select commands

### SELECT

b.book\_name, b.author,

COUNT(m.member\_id) AS member\_count FROM

Book1 b LEFT JOIN

Member1 m ON b.book\_id = m.member\_id WHERE

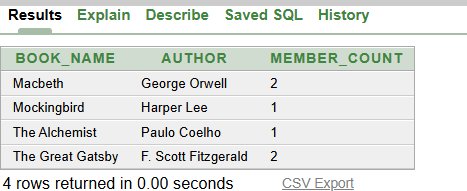
b.publication\_year >= 1925 GROUP BY

b.book\_name, b.author HAVING

COUNT(m.member\_id) > 0 ORDER BY

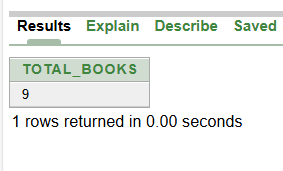
b.book\_name ASC;

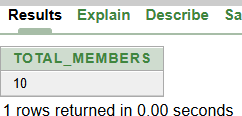
**Output:**



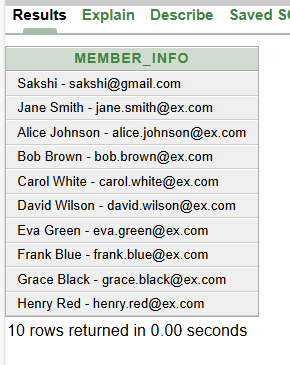
## Que.Write SQL Statement by using Oracle functions.

1. **Count the Number of Books:**

SELECT COUNT(\*) AS Total\_Books FROM Book1;

SELECT COUNT(\*) AS total\_members FROM Member1;

## Concatenate Name and Email of Members

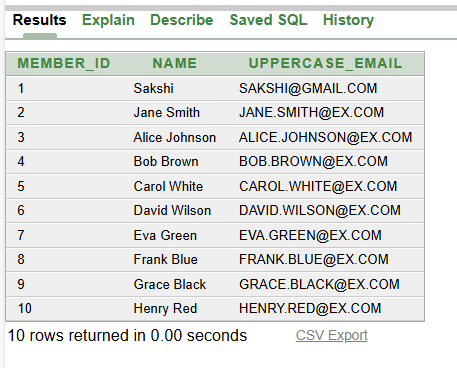
SELECT name || ' - ' || email AS member\_info FROM Member1;

1. **Get members whose names start with a specific letter (e.g., 'A')**: SELECT \* FROM Member1 WHERE UPPER(name) LIKE 'A%';



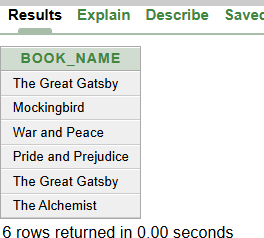
## Retrieve Members with Emails in Uppercase

SELECT Member\_id, Name, UPPER(Email) AS Uppercase\_Email FROM Member1;



## List Book Titles with Length Greater Than 10 Characters

SELECT Book\_name FROM Book1 WHERE LENGTH(Book\_name) > 10;



# ASSIGNMENT :6

## Que.Create view by using table book an member perform select ,update,delete operation

CREATE VIEW BookMemberView AS SELECT

B.Book\_id, B.book\_name, B.author,

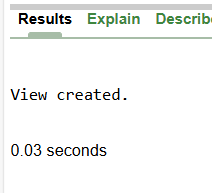
B.publication\_year, M.Member\_id, M.name,

M.email

FROM Book1 B JOIN

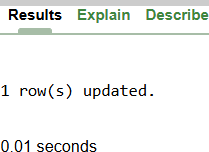
Member1 M ON B.Book\_id = M.Member\_id;

**Output:**



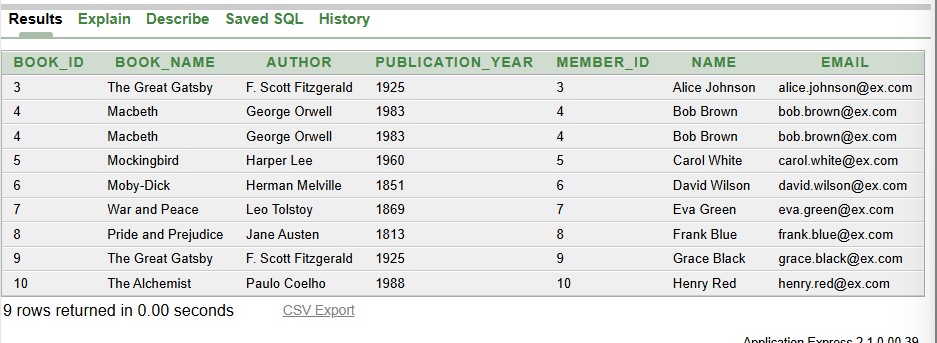
## Update Operation

UPDATE Member1 SET Email = 'sakshi@gmail.com' WHERE Member\_id = 1;



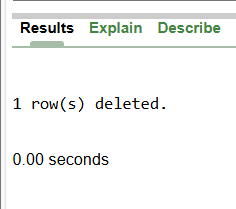
## Select operation

SELECT \* FROM BookMemberView;



## Delete Operation:

DELETE FROM Book1 WHERE Book\_id = 3;



## Que. write a sub query for the book and member table for library

* 1. **Sub Query**

SELECT name, email FROM member

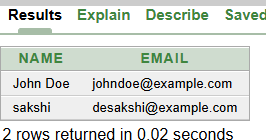
WHERE member\_id IN ( SELECT member\_id

FROM Book

WHERE Book\_id = (SELECT Book\_id FROM book WHERE publication\_year=2021)

);

## Output:

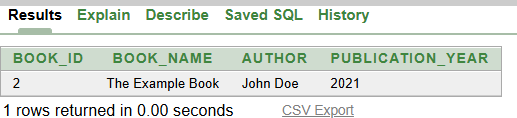


* 1. **Sub Query**

SELECT book\_id, title, author, publication\_year FROM Book

WHERE author IN (SELECT name FROM Member);

## Output:



**Que. write a PL/SQL by using if statement to display squares of all odd numbers between 1 to 25**

### BEGIN

FOR i IN 1..25 LOOP

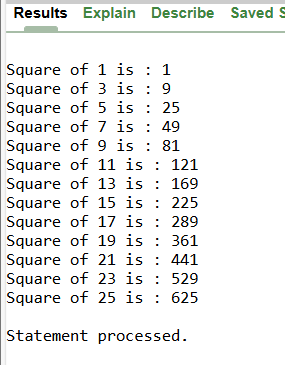
-- Check if the number is odd IF MOD(i, 2) = 1 THEN

-- Display the square of the odd number

DBMS\_OUTPUT.PUT\_LINE('Square of ' || i || ' is : ' || i \* i); END IF;

END LOOP; END;

**Output:**



**Que. write a PL/SQl to implement while loop for library table.**

1. **Library Table**

CREATE TABLE library (

book\_id NUMBER PRIMARY KEY, title VARCHAR2(255),

author VARCHAR2(255), available\_copies NUMBER

);

## Insert Into Library

INSERT INTO library (book\_id, title, author, available\_copies) VALUES (1, '1984', 'George Orwell', 5);

INSERT INTO library (book\_id, title, author, available\_copies) VALUES (2, 'To Kill a Mockingbird', 'Harper Lee', 3);

## While loop

### DECLARE

v\_book\_id NUMBER; v\_title VARCHAR2(255);

v\_author VARCHAR2(255); v\_available\_copies NUMBER;

CURSOR c\_library IS

SELECT book\_id, title, author, available\_copies FROM library

WHERE available\_copies > 0;

v\_counter NUMBER := 0; BEGIN

OPEN c\_library;

### LOOP

FETCH c\_library INTO v\_book\_id, v\_title, v\_author, v\_available\_copies; EXIT WHEN c\_library%NOTFOUND; -- Exit condition for the loop

-- Process the fetched data (in this case, print it)

DBMS\_OUTPUT.PUT\_LINE('Book ID: ' || v\_book\_id || ', Title: ' || v\_title ||

', Author: ' || v\_author ||

', Available Copies: ' || v\_available\_copies);

v\_counter := v\_counter + 1; END LOOP;

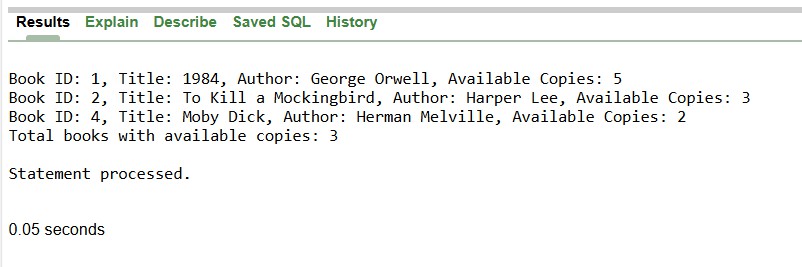
CLOSE c\_library;

-- Optional: Print the total count of books processed

DBMS\_OUTPUT.PUT\_LINE('Total books with available copies: ' || v\_counter); END;

/

## Output:



**Que. write a PL/SQl to implement For loop for library table**

### DECLARE

-- Declare a cursor to fetch data from the library table CURSOR library\_cursor IS

SELECT book\_id, title, author, available\_copies FROM library;

### BEGIN

-- Loop through each record fetched by the cursor FOR book\_record IN library\_cursor LOOP

-- Output the details of each book

DBMS\_OUTPUT.PUT\_LINE('Book ID: ' || book\_record.book\_id || ', Title: ' || book\_record.title ||

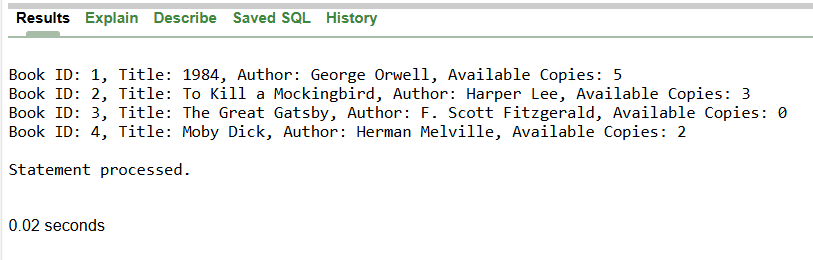
', Author: ' || book\_record.author ||

', Available Copies: ' || book\_record.available\_copies);

END LOOP;

END;

## Output:



**Que.Implement Error handling in Pl/SQL for library Book Table.**

### DECLARE

v\_Book\_id NUMBER := 101; -- Example book ID

v\_Book\_name VARCHAR2(100) := 'The Great Gatsby'; -- Example title v\_Author VARCHAR2(100) := 'F. Scott Fitzgerald'; -- Example author

v\_Publication\_Year NUMBER := 1925; -- Example year BEGIN

-- Attempt to insert a new book record

INSERT INTO Book (Book\_id, Book\_name, Author, Publication\_Year) VALUES (v\_Book\_id, v\_Book\_name, v\_Author, v\_Publication\_Year);

DBMS\_OUTPUT.PUT\_LINE('Book record inserted successfully.'); EXCEPTION

### WHEN DUP\_VAL\_ON\_INDEX THEN

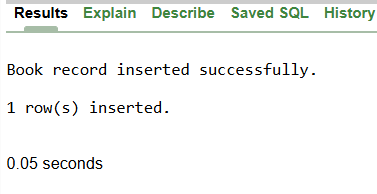
DBMS\_OUTPUT.PUT\_LINE('Error: Book ID already exists.');

### WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Invalid value for one of the fields.'); WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred: ' || SQLERRM); END**;**

**Output:**



## Que. write a PL/SQl to implement Procedure For the Given Table

CREATE OR REPLACE PROCEDURE insert\_book\_and\_member ( p\_book\_id IN NUMBER,

p\_book\_name IN VARCHAR2, p\_author IN VARCHAR2,

p\_publication\_year IN NUMBER, p\_member\_id IN NUMBER, p\_name IN VARCHAR2,

p\_email IN VARCHAR2

### ) AS BEGIN

-- Insert into the Book Table

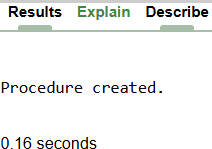
INSERT INTO Book (Book\_ID, book\_name, Author, publication\_year) VALUES (p\_book\_id, p\_book\_name, p\_author, p\_publication\_year);

-- Insert into the Member Table

INSERT INTO Member (Member\_ID, Name, email) VALUES (p\_member\_id, p\_name, p\_email);

-- Commit the transaction COMMIT;

### END;

**Output:**

### BEGIN

insert\_book\_and\_member( p\_book\_id => 1,

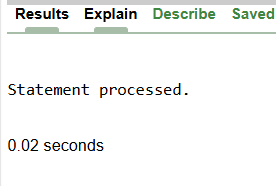
p\_book\_name => 'The Great Gatsby', p\_author => 'F. Scott Fitzgerald', p\_publication\_year => 1925,

p\_member\_id => 101, p\_name => 'John Doe',

p\_email => 'johndoe@example.com'

### ); END;

**Output:**



# ASSIGNMENT :13

## Que. write a PL/SQl to create and manage cursor for Book Table

### DECLARE

-- Cursor to fetch Book table details CURSOR book\_cursor IS

SELECT Book\_ID, book\_name, Author, publication\_year FROM Book;

-- Variables to hold book details fetched from the cursor v\_book\_id Book.Book\_ID%TYPE;

v\_book\_name Book.book\_name%TYPE; v\_author Book.Author%TYPE;

v\_publication\_year Book.publication\_year%TYPE;

### BEGIN

-- Open the cursor OPEN book\_cursor;

-- Loop through each row fetched by the cursor LOOP

FETCH book\_cursor INTO v\_book\_id, v\_book\_name, v\_author, v\_publication\_year;

-- Exit the loop when no more rows are fetched EXIT WHEN book\_cursor%NOTFOUND;

-- Display the book details

DBMS\_OUTPUT.PUT\_LINE('Book ID: ' || v\_book\_id || ', Name: ' || v\_book\_name || ', Author: ' || v\_author || ', Year: ' || v\_publication\_year);

### END LOOP;

-- Close the cursor CLOSE book\_cursor; END;

**Output:**

