IFSC 3330

LIBRARY MANAGEMENT SYSTEM

(LMS)

PARIKSHYA BHANDARI

PROFESSOR NINGNING WU

DATE: APRIL 25, 2024

INTRODUCTION:

The Library Management System (LMS) project, designed for IFSC 3330: Current Trends in Database Technology, is a significant advancement in digitizing and enhancing library operations. The project leverages advanced database technology to streamline processes, aiming to boost operational efficiency and user satisfaction. This system is set to enhance library resource management and service delivery, improving accessibility and engagement for both staff and patrons.

In the first phase, the development centered on creating an Entity-Relationship (ER) diagram, outlining key entities such as Staff, Readers, Books, and Publishers, and their interactions like Reserves and Returns. This stage was crucial for setting up a robust database design for efficient data management.

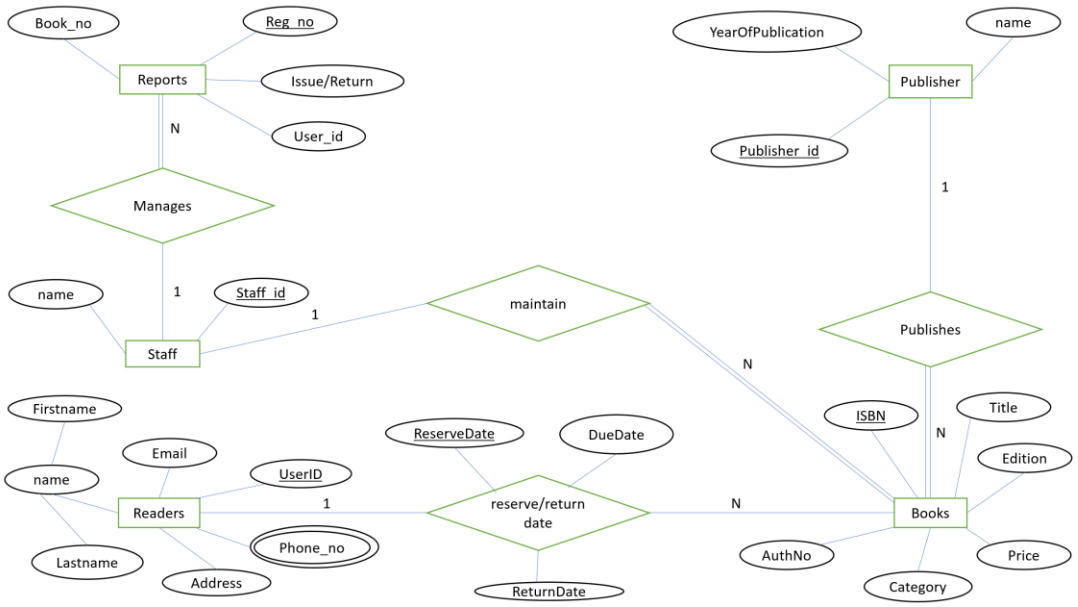
The second phase addressed the normalization of the database schema, transitioning from the First Normal Form (1NF) to overcome issues like partial and transitive dependencies. The database was organized into distinct relations—Reports, Staff, Books, Reserve/Return, Readers, and Publisher—to manage different aspects of library operations effectively.

The third phase involved setting up the actual database using SQL to build tables and manage data interactions through insertions and updates. This setup included establishing foreign key constraints for data integrity and populating tables with initial data, thus preparing the system for real-world use.

Finally, the last phase focused on deploying SQL queries for data retrieval and manipulation according to user needs, including generating reports and handling book transactions. The introduction of triggers and stored procedures automated routine tasks, ensuring the system remained current and responsive to library needs.

PROJECT PHASE 1:

ER Diagram:



Entities and Attributes:

1. **Staff:**
   - Staff_id (Primary Key)
   - Name
2. **Readers:**
   - User_ID (Primary Key)
   - Firstname
   - LastName
   - Email
   - Phone_no (MVA)
   - Address

3. **Books**:
   - ISBN (Primary Key)
   - AuthNo
   - Title
   - Edition
   - Category
   - Price
4. **Publisher:**
   - Publisher_id (Primary Key)
   - Name
   - YearOfPublication
5. **Reserve/Return (Relationship Set):**
   - ReserveDate (Primary Key)
   - return_date
   - Due date
6. **Reports:**
   - Reg_no (Primary Key)
   - Book_No
   - User_id
   - Issue/return date

Relationships:

1. Reader-Book: A reader can reserve N books, but one book can be reserved by only one reader. The relationship 1:N.
2. Publisher-Book: A publisher can publish many books, but a book is published by only one publisher. The relationship 1:N.
3. Staff-Reports: Staff maintains multiple reports. The relationship 1:N.
4. Staff-Books: Staffs maintain multiple books. The relationship 1:N.

Assumptions:

1. Each Staff member has a unique Staff_id.
2. Each Reader has a unique User_ID.
3. Each Book has a unique ISBN.
4. Each Publisher has a unique Publisher_id.
5. Each Report has a unique Reg_no.

Participation Constraints:

- In the Reader-Book relationship, every Book must be reserved by at least one Reader (total participation of Book in the relationship).
- In the Publisher-Book relationship, every Book must be published by exactly one Publisher (total participation of Publisher in the relationship).
- In the Staff-Reports relationship, every Report must be maintained by at least one Staff (total participation of Staff in the relationship).
- In the Staff-Books relationship, every Book must be maintained by at least one Staff (total participation of Staff in the relationship).

Design Goals:

- Ensure efficient management of staff, readers, books, publishers, and reports within the system.
- Allow readers to reserve multiple books while ensuring each book is reserved by only one reader at a time.
- Enable publishers to manage their published books effectively.
- Facilitate staff members in maintaining and generating reports related to book transactions and reader interactions.

PROJECT PHASE 2:

LibraryManagementSystem (Reg_no, Book_no, Issue/Return, User_id, Staff_id, Staff_name, ISBN, Title, Edition, Price, Category, AuthNo, ReserveDate, ReturnDate, DueDate, UserID, Firstname, Lastname, Email, Address, Phone_no, Publisher_id, Publisher_name, YearOfPublication)

FD1: Reg_no --> Book_no, Issue/Return, User_id, Staff_id

FD2: Staff_id --> Staff_name

FD3: ISBN --> Title, Edition, Price, Category, AuthNo, Publisher_id, Staff_id

FD4: ReserveDate --> ReturnDate, DueDate, UserID, ISBN

FD5: UserID --> Firstname, Lastname, Email, Address, Phone_no

FD6: Publisher_id --> Publisher_name, YearOfPublication

Primary Attributes: Reg_no, Staff_id, ISBN, UserID, Publisher_id, ReserveDate
Non-primary Attributes: Book_no, Issue/Return, User_id, Staff_name, Title, Edition, Price, Category, AuthNo, ReturnDate, DueDate, Firstname, Lastname, Email, Address, Phone_no, Publisher_name, YearOfPublication
Partial FDs (if any): FD1, FD4
Transitive FDs (if any): FD2, FD3, FD5, FD6
Full FDs (if any): None

LibraryManagementSystem is only in **1NF** because all attributes are atomic, meaning each attribute value must be indivisible.

LibraryManagementSystem is not in **2NF** or **3NF** due to the presence of partial and transitive dependencies.

Step 1:

- **FD1:** Reg_no --> Book_no, Issue/Return, User_id, Staff_id **(partial)**
- **FD2:** Staff_id --> Staff_name **(transitive)**
- **FD3:** ISBN --> Title, Edition, Price, Category, AuthNo, Publisher_id, Staff_id **(transitive)**
- **FD4:** ReserveDate --> ReturnDate, DueDate, UserID, ISBN **(partial)**
- **FD5:** UserID --> Firstname, Lastname, Email, Address, Phone_no **(transitive)**
- **FD6:** Publisher_id --> Publisher_name, YearOfPublication **(transitive)**

Decomposed Relations:

Reports (Reg_no, Book_no, Issue/Return, User_id, Staff_id)

Staff (Staff_id, Staff_name)

Books (ISBN, Title, Edition, Price, Category, AuthNo, Publisher_id, Staff_id)

Reserve/Return (ReserveDate, DueDate, ReturnDate, UserID, ISBN)

Readers (UserID, Firstname, Lastname, Email, Address, Phone_no)

Publisher (Publisher_id, Publisher_name, YearOfPublication)

Step2: (creating a schema that has the primary key of LMS and attributes that are fully dependent on it (if any).)

There are no full FDs. So, this step is skipped.

Final Results:

Reports (Reg_no, Book_no, Issue/Return, User_id, Staff_id)

Staff (Staff_id, Staff_name)

Books (ISBN, Title, Edition, Price, Category, AuthNo, Publisher_id, Staff_id)

Reserve/Return (ReserveDate, DueDate, ReturnDate, UserID, ISBN)

Readers (UserID, Firstname, Lastname, Email, Address, Phone_no)

Publisher (Publisher_id, Publisher_name, YearOfPublication)

Foreign Key Notations:

- Reports.Staff_id --> Staff.Staff_id
- Books.Staff_id --> Staff.Staff_id
- Books.Publisher_id --> Publisher.Publisher_id
- Reserve/Return.ISBN --> Books.ISBN
- Reserve/Return.UserID --> Readers.UserID

PROJECT PHASE 3:

-- *Reports table*

```
DROP TABLE Reports CASCADE CONSTRAINTS;

CREATE TABLE Reports (
    Reg_no INT PRIMARY KEY,
    Book_no VARCHAR(20),
    IssueReturn VARCHAR(10),
    User_id INT,
    Staff_id INT,
    FOREIGN KEY (Staff_id) REFERENCES Staff(Staff_id)
);

INSERT INTO Reports (Reg_no, Book_no, IssueReturn, User_id, Staff_id) VALUES (001, 'ISBN123456', 'Issue', 101, 1);
INSERT INTO Reports (Reg_no, Book_no, IssueReturn, User_id, Staff_id) VALUES (002, 'ISBN234567', 'Return', 102, 2);
INSERT INTO Reports (Reg_no, Book_no, IssueReturn, User_id, Staff_id) VALUES (003, 'ISBN345678', 'Issue', 103, 3);
```

-- *Staff table*

```
DROP TABLE Staff CASCADE CONSTRAINTS;

CREATE TABLE Staff (
    Staff_id INT PRIMARY KEY,
```

```
    Staff_name VARCHAR(50)
);
INSERT INTO Staff (Staff_id, Staff_name) VALUES (1, 'John
Doe');
INSERT INTO Staff (Staff_id, Staff_name) VALUES (2, 'Jane
Smith');
INSERT INTO Staff (Staff_id, Staff_name) VALUES (3, 'Michael
Johnson')

-- Books table
DROP TABLE Books CASCADE CONSTRAINTS;
CREATE TABLE Books (
    ISBN VARCHAR(20) PRIMARY KEY,
    VARCHAR(100),
    Edition VARCHAR(50),
    Price DECIMAL(10,2),
    Category VARCHAR(50),
    AuthNo INT,
    Publisher_id INT,
    Staff_id INT,
    FOREIGN KEY (Staff_id) REFERENCES Staff(Staff_id),
    FOREIGN KEY (Publisher_id) REFERENCES Publisher(Publisher_id)
```

```
);

INSERT INTO Books (ISBN, Title, Edition, Price, Category, AuthNo, Publisher_id, Staff_id) VALUES ('ISBN123456', 'Introduction to SQL', '2nd Edition', 45.99, 'Database', 1234, 1, 1);

INSERT INTO Books (ISBN, Title, Edition, Price, Category, AuthNo, Publisher_id, Staff_id) VALUES ('ISBN234567', 'Python Programming', '3rd Edition', 39.99, 'Programming', 2345, 2, 2);

INSERT INTO Books (ISBN, Title, Edition, Price, Category, AuthNo, Publisher_id, Staff_id) VALUES ('ISBN345678', 'Data Structures and Algorithms', '4th Edition', 55.99, 'Computer Science', 3456, 3, 3);


-- Reserve/Return table

DROP TABLE ReserveReturn CASCADE CONSTRAINTS;

CREATE TABLE ReserveReturn (

    ReserveDate DATE PRIMARY KEY,

    DueDate DATE,

    ReturnDate DATE,

    UserID INT,

    ISBN VARCHAR(20),

    FOREIGN KEY (UserID) REFERENCES Readers(UserID),

    FOREIGN KEY (ISBN) REFERENCES Books(ISBN)

);
```

INSERT INTO ReserveReturn (ReserveDate, DueDate, ReturnDate, UserID, ISBN) **VALUES** (TO_DATE('2024-03-01', 'YYYY-MM-DD'), TO_DATE('2024-03-15', 'YYYY-MM-DD'), NULL, 101, 'ISBN123456');

INSERT INTO ReserveReturn (ReserveDate, DueDate, ReturnDate, UserID, ISBN) **VALUES** (TO_DATE('2024-03-05', 'YYYY-MM-DD'), TO_DATE('2024-03-20', 'YYYY-MM-DD'), NULL, 102, 'ISBN234567');

INSERT INTO ReserveReturn (ReserveDate, DueDate, ReturnDate, UserID, ISBN) **VALUES** (TO_DATE('2024-03-10', 'YYYY-MM-DD'), TO_DATE('2024-03-25', 'YYYY-MM-DD'), NULL, 103, 'ISBN345678');

-- *Readers table*

DROP TABLE **Readers** CASCADE CONSTRAINTS;
CREATE TABLE Readers (
    UserID INT **PRIMARY KEY**,
    Firstname VARCHAR(50),
    Lastname VARCHAR(50),
    Email VARCHAR(100),
    Address VARCHAR(200),
    Phone_no VARCHAR(15)
);

INSERT INTO Readers (UserID, Firstname, Lastname, Email, Address, Phone_no) **VALUES**
(101, 'Alice', 'Johnson', 'alice@example.com', '123 Main St, City', '123-456-7890');

INSERT INTO Readers (UserID, Firstname, Lastname, Email, Address, Phone_no) **VALUES**
(102, 'Bob', 'Smith', 'bob@example.com', '456 Elm St, Town', '456-789-0123');

INSERT INTO Readers (UserID, Firstname, Lastname, Email, Address, Phone_no)
VALUES
(103, 'Charlie', 'Brown', 'charlie@example.com', '789 Oak St, Village', '789-012-3456');

-- *Publisher table*
DROP TABLE **Publisher** CASCADE CONSTRAINTS;

CREATE TABLE Publisher (
Publisher_id INT **PRIMARY KEY**,
Publisher_name VARCHAR(100),
YearOfPublication INT
);

INSERT INTO Publisher (Publisher_id, Publisher_name, YearOfPublication)
VALUES (1, 'Penguin Books', 2005);

INSERT INTO Publisher (Publisher_id, Publisher_name, YearOfPublication)
VALUES (2, 'HarperCollins', 2010);

INSERT INTO Publisher (Publisher_id, Publisher_name, YearOfPublication)
VALUES (3, 'Random House', 2015);

PROJECT PHASE 4:

Queries:

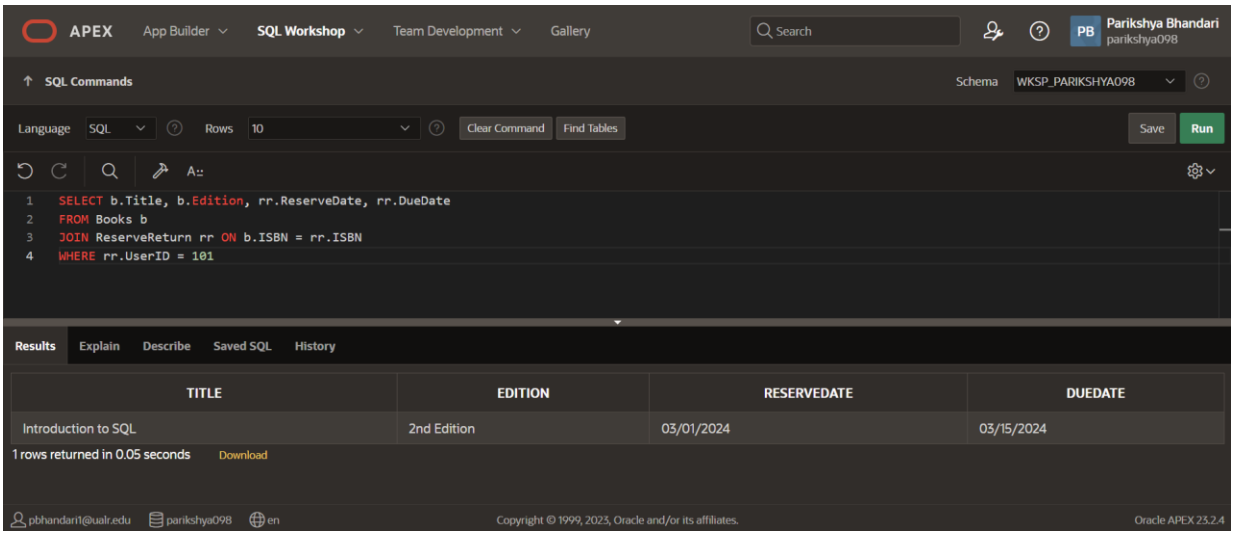
1. List All Books Issued by a Specific User:

SELECT b.Title, b.Edition, rr.ReserveDate, rr.DueDate

FROM Books b

JOIN ReserveReturn rr ON b.ISBN = rr.ISBN

WHERE rr.UserID = 101



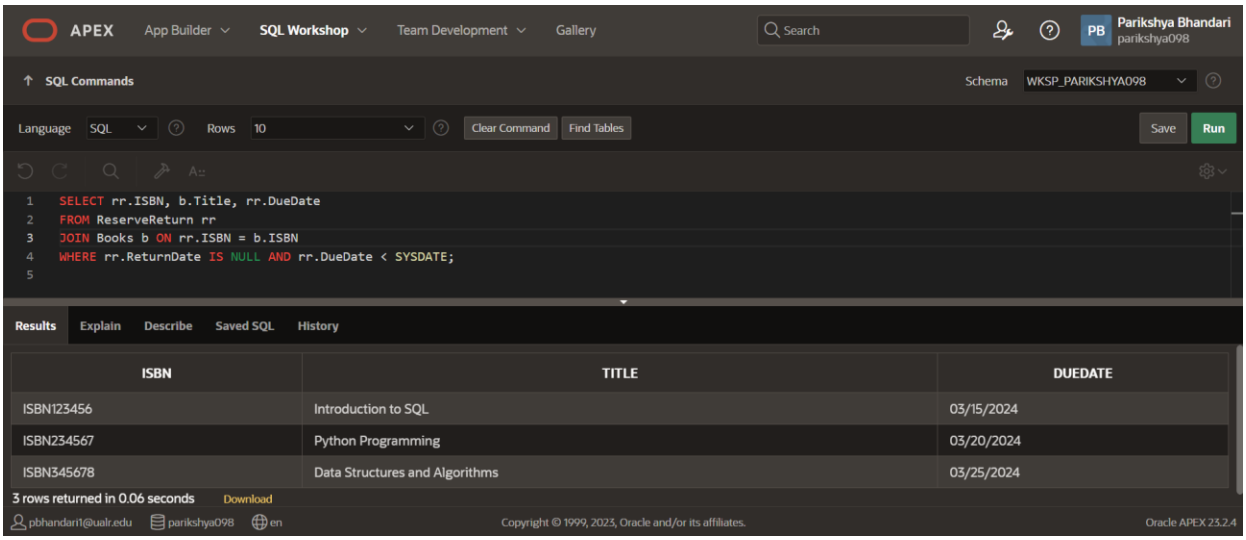
2. Find Overdue Books:

SELECT rr.ISBN, b.Title, rr.DueDate

FROM ReserveReturn rr

JOIN Books b ON rr.ISBN = b.ISBN

WHERE rr.ReturnDate IS NULL AND rr.DueDate < SYSDATE;



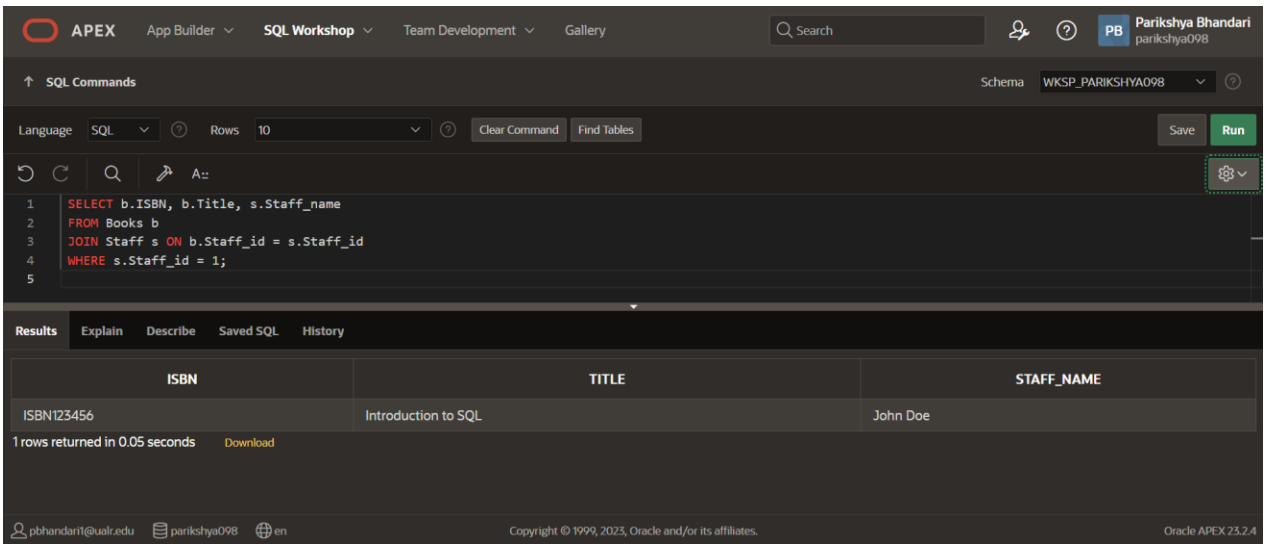
3. List Books Handled by a Specific Staff Member:

SELECT b.ISBN, b.Title, s.Staff_name

FROM Books b

JOIN Staff s ON b.Staff_id = s.Staff_id

WHERE s.Staff_id = 1;



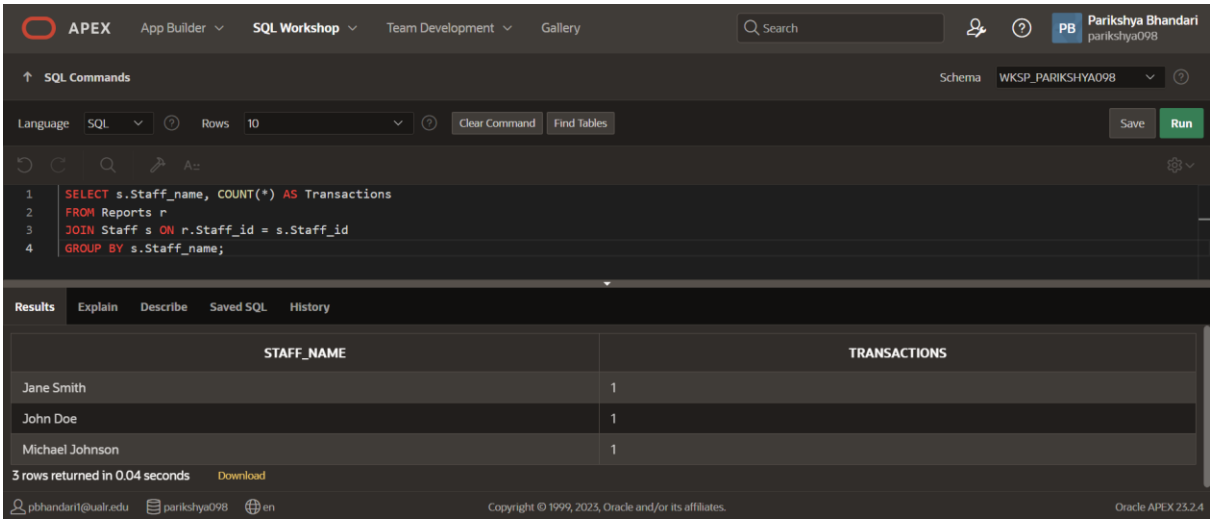
4. Count of Books Issued and Returned by Each Staff Member:

SELECT s.Staff_name, COUNT(*) AS Transactions

FROM Reports r

JOIN Staff s ON r.Staff_id = s.Staff_id

GROUP BY s.Staff_name;



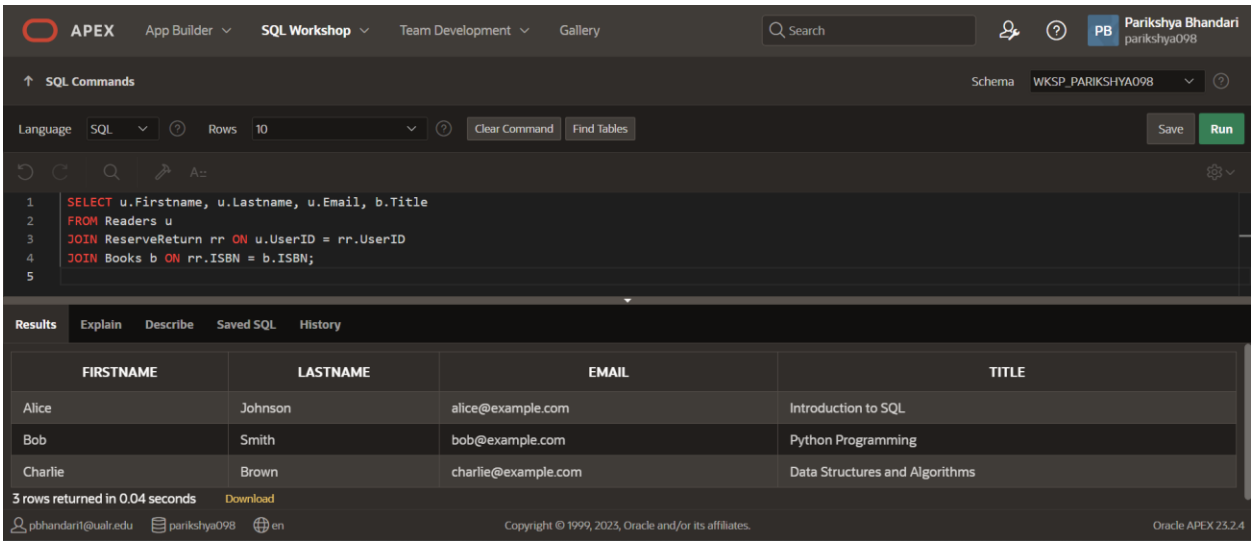
5. Detailed User Information Including Books They Reserved:

SELECT u.Firstname, u.Lastname, u.Email, b.Title

FROM Readers u

JOIN ReserveReturn rr ON u.UserID = rr.UserID

JOIN Books b ON rr.ISBN = b.ISBN;



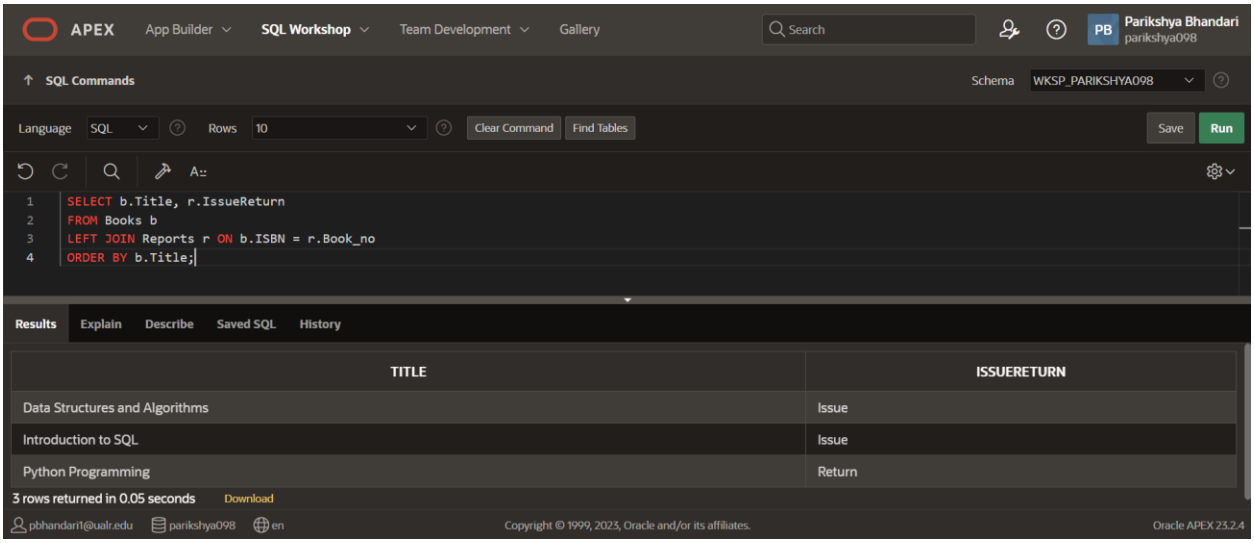
6. List All Books with Their Current Status (Issued/Returned):

SELECT b.Title, r.IssueReturn

FROM Books b

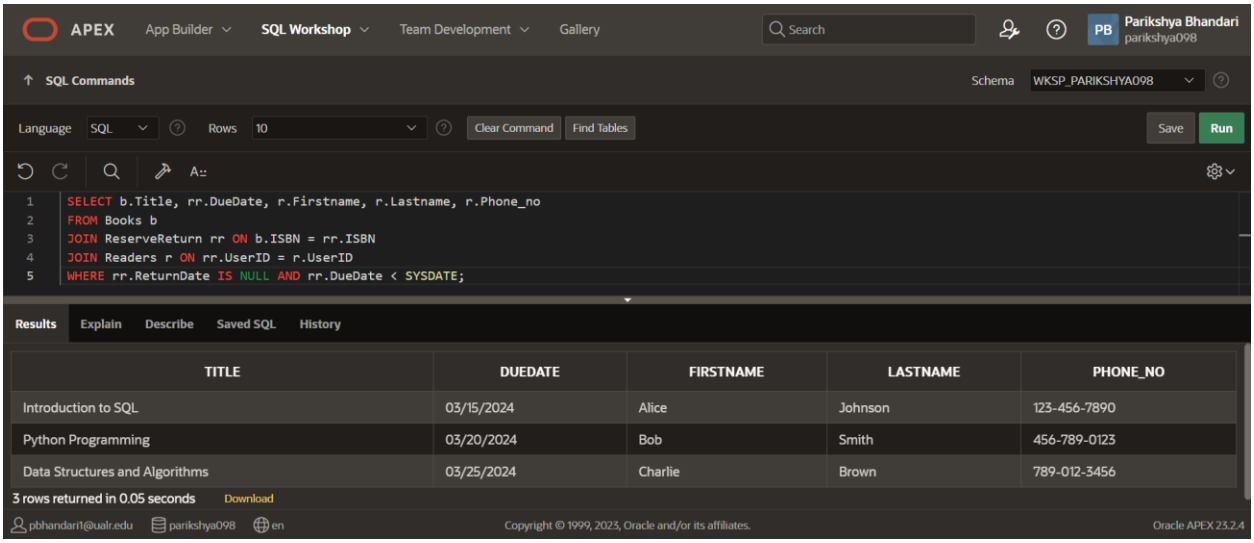
LEFT JOIN Reports r ON b.ISBN = r.Book_no

ORDER BY b.Title;



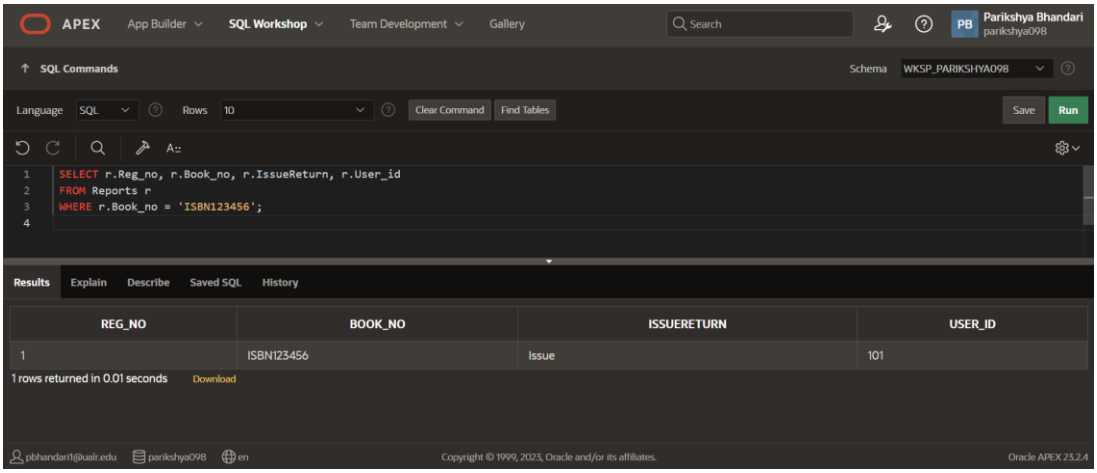
7. Report Overdue Books with User Contact Information:

```
SELECT b.Title, rr.DueDate, r.Firstname, r.Lastname, r.Phone_no

FROM Books b

JOIN ReserveReturn rr ON b.ISBN = rr.ISBN

JOIN Readers r ON rr.UserID = r.UserID

WHERE rr.ReturnDate IS NULL AND rr.DueDate < SYSDATE;
```



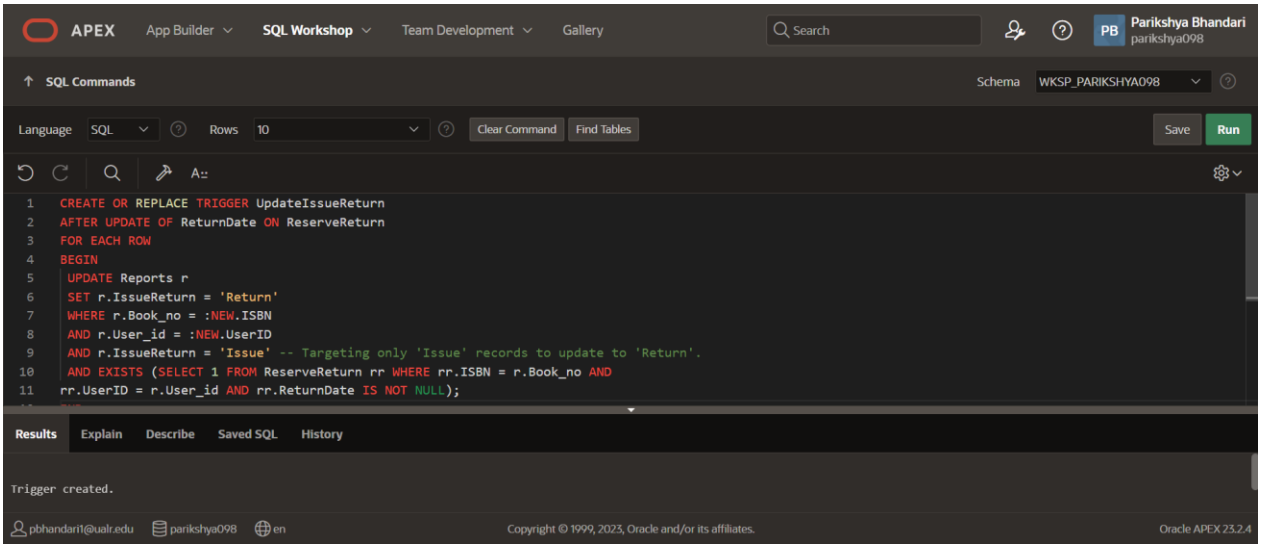
8. History of a Book (Issues and Returns):

```
SELECT r.Reg_no, r.Book_no, r.IssueReturn, r.User_id

FROM Reports r

WHERE r.Book_no = 'ISBN123456';
```



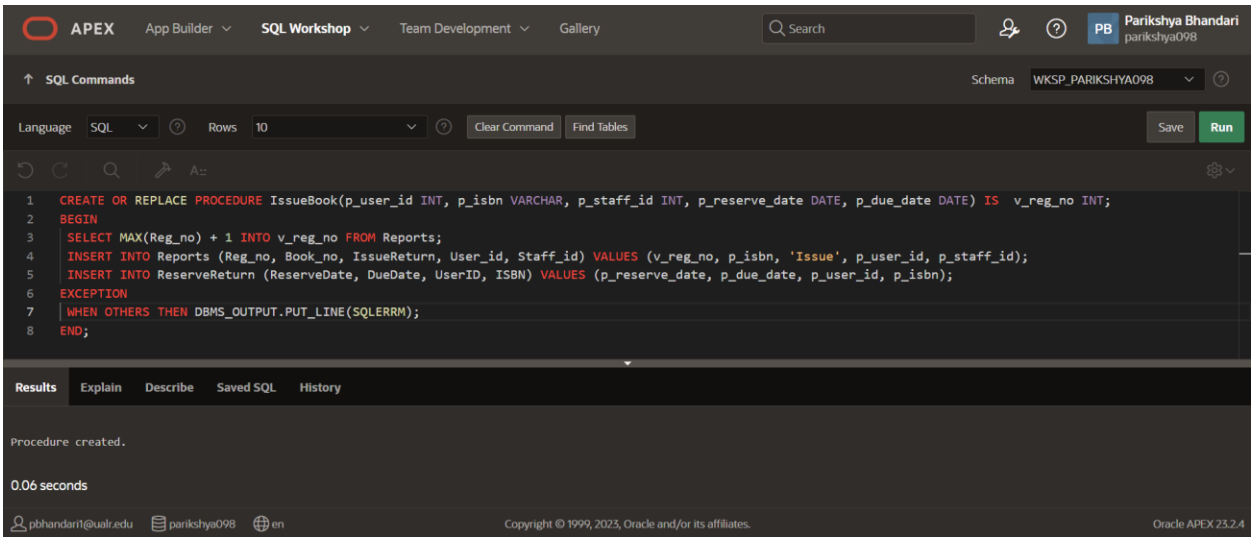
Trigger: Update Status on Book Return

```
CREATE OR REPLACE TRIGGER UpdateIssueReturn

AFTER UPDATE OF ReturnDate ON ReserveReturn

FOR EACH ROW

BEGIN

    UPDATE Reports r

    SET r.IssueReturn = 'Return'

    WHERE r.Book_no = :NEW.ISBN

    AND r.User_id = :NEW.UserID

    AND r.IssueReturn = 'Issue' -- Targeting only 'Issue' records to update to 'Return'.

    AND EXISTS (SELECT 1 FROM ReserveReturn rr WHERE rr.ISBN =
    r.Book_no AND rr.UserID = r.User_id AND rr.ReturnDate IS NOT NULL);

END;
```



Procedures or Functions: Procedure to Issue a Book

```
CREATE OR REPLACE PROCEDURE IssueBook(p_user_id INT, p_isbn VARCHAR, p_staff_id INT, p_reserve_date DATE, p_due_date DATE) IS  v_reg_no INT;

BEGIN

    SELECT MAX(Reg_no) + 1 INTO v_reg_no FROM Reports;

    INSERT INTO Reports (Reg_no, Book_no, IssueReturn, User_id, Staff_id)
    VALUES (v_reg_no, p_isbn, 'Issue', p_user_id, p_staff_id);

    INSERT INTO ReserveReturn (ReserveDate, DueDate, UserID, ISBN) VALUES
    (p_reserve_date, p_due_date, p_user_id, p_isbn);

EXCEPTION

    WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);

END;
```



CONCLUSION:

The development of the Library Management System through these phases represents a significant enhancement in library management and user interaction. By transitioning from traditional methods to a sophisticated database system, the project not only improves operational efficiency but also enhances user engagement. As libraries continue to serve as vital resources for information and learning, the LMS ensures they remain relevant and accessible in the digital age, aligning with current trends in database technology and setting a foundation for future enhancements.