INFORMATION SYSTEM 3

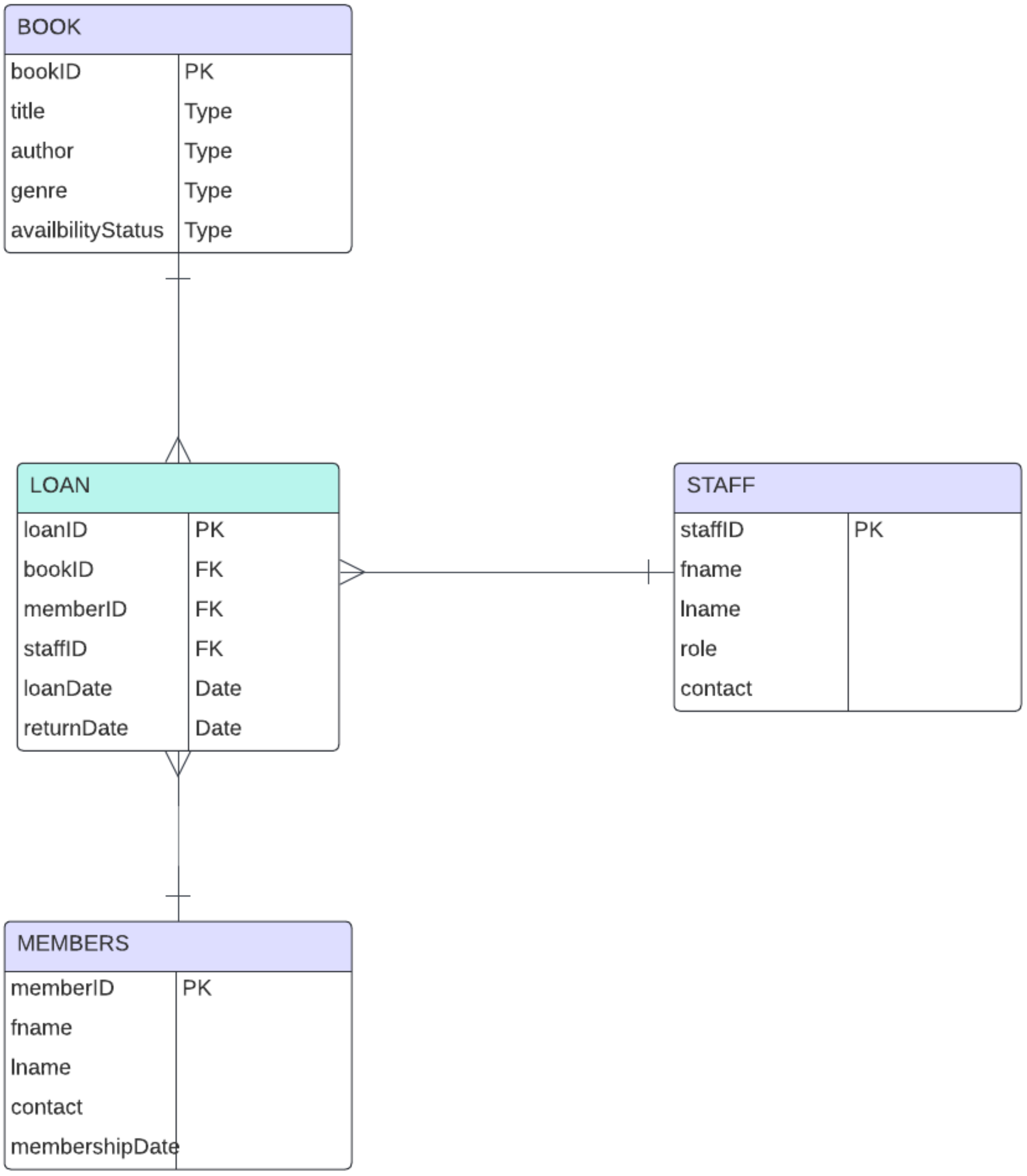
Assignment 1: Library Management System

Presented by: **Onpoint.**

NAME	CONTECT DETAILS	PERCENTILE CONTRIBUTION	SIGNITURE
S. Khanyile 22211204	22211204@live.mut.ac.za	100%	[illegible]
B. Zulu 22202710	22202710@live.mut.ac.za	22202710@live.mut.ac.za	[illegible]
SA. Mnyandu 22222562	22222562@live.mut.ac.za	100%	[illegible]

Contents

Question 1


BOOK
bookID
PK
title
Type
author
Type
genre
Type
availbilityStatus
Type
LOAN
loanID
PK
bookID
FK
memberID
FK
staffID
FK
loanDate
Date
returnDate
Date
STAFF
staffID
PK
fname
lname
role
contact
MEMBERS
memberID
PK
fname
lname
contact
membershipDate

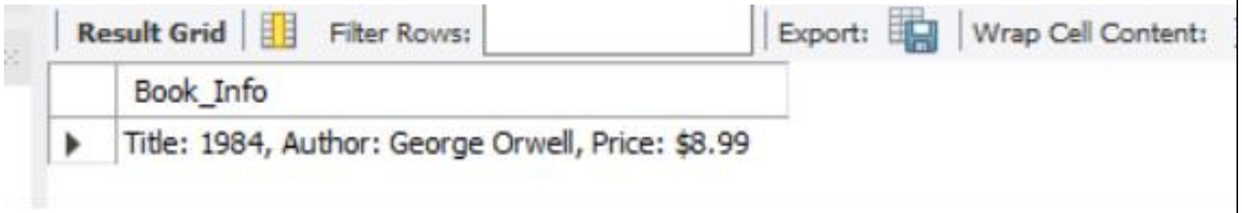
Question 2

```
-- Create Books Table
CREATE TABLE Books (
   BookID INT PRIMARY KEY,
   Title VARCHAR(255),
   Author VARCHAR(255),
   Year INT,
   Price DECIMAL(5, 2)
);

-- Create Members Table
CREATE TABLE Members (
   MemberID INT PRIMARY KEY,
   FirstName VARCHAR(255),
   LastName VARCHAR(255),
   PhoneNumber VARCHAR(15),
   Address VARCHAR(255)
);

-- Create Loans Table
CREATE TABLE Loans (
   LoanID INT PRIMARY KEY,
   BookID INT,
   MemberID INT,
   LoanDate DATE,
   ReturnDate DATE,
   FOREIGN KEY (BookID) REFERENCES Books(BookID),
   FOREIGN KEY (MemberID) REFERENCES Members(MemberID)
);

-- Populate Books Table
INSERT INTO Books (BookID, Title, Author, Year, Price) VALUES
(101, 'The Great Gatsby', 'F. Scott', 1925, 10.99),
(102, '1984', 'George Orwell', 1949, 8.99),
(103, 'To Kill a Mockingbird', 'Harper Lee', 1960, 12.99);

-- Populate Members Table
INSERT INTO Members (MemberID, FirstName, LastName, PhoneNumber, Address) VALUES
(201, 'John', 'Doe', '123-456-7890', '123 Main St'),
(202, 'Jane', 'Smith', '987-654-3210', '456 Elm St'),
(203, 'Emily', 'Johnson', '555-555-5555', '789 Oak St');

-- Populate Loans Table
```

```
INSERT INTO Loans (LoanID, BookID, MemberID, LoanDate, ReturnDate) VALUES
(301, 101, 201, '2023-06-01', '2023-06-15'),
(302, 102, 202, '2023-06-05', '2023-06-20'),
(303, 103, 203, '2023-06-10', '2023-06-25');
```

Question 3

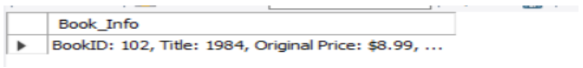
```
BEGIN
  FOR rec IN (SELECT Title, Author, Price
        FROM Books
        WHERE Price < 10) LOOP
    DBMS_OUTPUT.PUT_LINE('Title: ' || rec.Title || ', Author: ' || rec.Author || ', Price: $' || rec.Price);
  END LOOP;
END;
```



```
Book_Info
Title: 1984, Author: George Orwell, Price: $8.99
```

Question 4

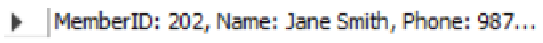
```
DECLARE
  CURSOR books_cursor IS
    SELECT BookID, Title, Price
    FROM Books
    WHERE Year < 1950;
  v_original_price Books.Price%TYPE;
  v_new_price Books.Price%TYPE;
BEGIN
  FOR rec IN books_cursor LOOP
    v_original_price := rec.Price;
    v_new_price := rec.Price * 1.15;

    UPDATE Books
    SET Price = v_new_price
    WHERE BookID = rec.BookID;

    DBMS_OUTPUT.PUT_LINE('BookID: ' || rec.BookID || ', Title: ' || rec.Title ||
               ', Original Price: $' || v_original_price ||
                  ', New Price: $' || v_new_price);
  END LOOP;
END;
```



```
Book_Info
BookID: 102, Title: 1984, Original Price: $8.99, ...
```

Question 5

```
DECLARE
  v_last_name Members.LastName%TYPE := 'Smith';
BEGIN
  FOR rec IN (SELECT m.MemberID, m.FirstName, m.LastName, m.PhoneNumber, m.Address,
b.Title
          FROM Members m
          JOIN Loans l ON m.MemberID = l.MemberID
          JOIN Books b ON l.BookID = b.BookID
          WHERE m.LastName = v_last_name) LOOP
      DBMS_OUTPUT.PUT_LINE('MemberID: ' || rec.MemberID || ', Name: ' || rec.FirstName || ' ' ||
rec.LastName ||
                  ', Phone: ' || rec.PhoneNumber || ', Address: ' || rec.Address ||
                  ', Borrowed Book: ' || rec.Title);
  END LOOP;
                    END;
```

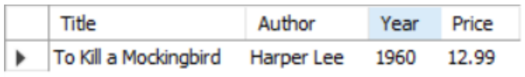


```
MemberID: 202, Name: Jane Smith, Phone: 987...
```

Question 6

```
CREATE OR REPLACE FUNCTION get_most_expensive_book RETURN Books%ROWTYPE IS
  v_book Books%ROWTYPE;
BEGIN
  SELECT * INTO v_book
  FROM Books
  ORDER BY Price DESC
  FETCH FIRST 1 ROW ONLY;

  RETURN v_book;
```

```
END;
```



	Title	Author	Year	Price
▶	To Kill a Mockingbird	Harper Lee	1960	12.99

Question 7

```
CREATE OR REPLACE PROCEDURE insert_loan (
  p_LoanID IN Loans.LoanID%TYPE,
  p_BookID IN Loans.BookID%TYPE,
  p_MemberID IN Loans.MemberID%TYPE,
  p_LoanDate IN Loans.LoanDate%TYPE,
  p_ReturnDate IN Loans.ReturnDate%TYPE
) IS
  e_duplicate EXCEPTION;
  PRAGMA EXCEPTION_INIT(e_duplicate, -1); -- Unique constraint violation
BEGIN
  INSERT INTO Loans (LoanID, BookID, MemberID, LoanDate, ReturnDate)
  VALUES (p_LoanID, p_BookID, p_MemberID, p_LoanDate, p_ReturnDate);
EXCEPTION
  WHEN e_duplicate THEN
    DBMS_OUTPUT.PUT_LINE('Error: Duplicate loan entry detected.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

Question 8

```
CREATE OR REPLACE FUNCTION get_member_id_by_last_name (
  p_last_name IN Members.LastName%TYPE
) RETURN Members.MemberID%TYPE IS
  v_member_id Members.MemberID%TYPE := 0;
BEGIN
  SELECT MemberID INTO v_member_id
  FROM Members
  WHERE LastName = p_last_name;

  RETURN v_member_id;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
```

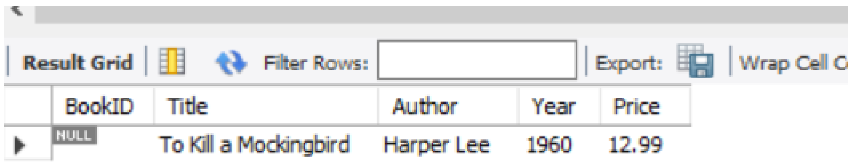
```
    RETURN 0;
END;
```

Question 9

```
-- Create LoansHistory Table
CREATE TABLE LoansHistory (
  HistoryID INT PRIMARY KEY,
  LoanID INT,
  BookID INT,
  MemberID INT,
  LoanDate DATE,
  ReturnDate DATE,
  Action VARCHAR2(10),
  ActionDate DATE
);

-- Create Sequence for LoansHistory
CREATE SEQUENCE LoansHistory_seq START WITH 1 INCREMENT BY 1;

-- Create Trigger
CREATE OR REPLACE TRIGGER trg_log_loans_changes
AFTER INSERT OR UPDATE OR DELETE ON Loans
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    INSERT INTO LoansHistory (HistoryID, LoanID, BookID, MemberID, LoanDate, ReturnDate,
Action, ActionDate)
    VALUES (LoansHistory_seq.NEXTVAL, :NEW.LoanID, :NEW.BookID, :NEW.MemberID,
:NEW.LoanDate, :NEW.ReturnDate, 'INSERT', SYSDATE);
  ELSIF UPDATING THEN
    INSERT INTO LoansHistory (HistoryID, LoanID, BookID, MemberID, LoanDate, ReturnDate,
Action, ActionDate)
    VALUES (LoansHistory_seq.NEXTVAL, :NEW.LoanID, :NEW.BookID, :NEW.MemberID,
:NEW.LoanDate, :NEW.ReturnDate, 'UPDATE', SYSDATE);
  ELSIF DELETING THEN
    INSERT INTO LoansHistory (HistoryID, LoanID, BookID, MemberID, LoanDate, ReturnDate,
Action, ActionDate)
    VALUES (LoansHistory_seq.NEXTVAL, :OLD.LoanID, :OLD.BookID, :OLD.MemberID,
:OLD.LoanDate, :OLD.ReturnDate, 'DELETE', SYSDATE);
  END IF;
END;
```

Question 10

```
CREATE OR REPLACE FUNCTION get_last_loan_date RETURN DATE IS
  v_last_loan_date DATE;
BEGIN
  SELECT MAX(LoanDate) INTO v_last_loan_date
  FROM Loans;

  RETURN v_last_loan_date;
END;
```



	BookID	Title	Author	Year	Price
▶	NULL	To Kill a Mockingbird	Harper Lee	1960	12.99