📝 SQL Comprehensive Examination

Course: Database Management Systems and SQL
Total Marks: 100
Duration: 2 Hours
Instructions:

- Answer ALL questions
- Each question carries 10 marks
- Write SQL queries clearly with proper formatting
- Provide explanations where asked
- Assume a Library Management System database with the following tables:
  - **Members** (MemberID, Name, Email, PhoneNumber, MemberType, JoinDate, City)
  - **Books** (BookID, Title, Author, Genre, Price, PublicationYear, ISBN, Quantity)
  - **Loans** (LoanID, MemberID, BookID, LoanDate, DueDate, ReturnDate, LateFee)
  - **Fines** (FineID, LoanID, MemberID, FineAmount, FineDate, IsPaid)

Question 1: DBMS Fundamentals & Data Types [10 Marks]

Part A (5 marks):
Explain the following concepts with examples:

1. What is RDBMS and how does it differ from a file system? (2 marks)
2. Define Primary Key and Foreign Key with an example from the Library System. (2 marks)
3. Why is normalization important? Name the three normal forms. (1 mark)

Part B (5 marks):
Create a table named `Staff` with the following specifications:

- StaffID (integer, primary key, auto-increment)
- FirstName (required, supports international characters, max 50 chars)
- LastName (required, supports international characters, max 50 chars)
- Email (unique, required, max 100 chars)
- PhoneNumber (optional, max 15 chars)
- Salary (decimal with 2 decimal places, must be positive)
- HireDate (date only)

- IsActive (boolean, default true)
- Department (optional, max 50 chars, default 'General')

Write the complete CREATE TABLE statement with all appropriate data types and constraints.

Question 2: DDL, DML, DQL, TCL, DCL [10 Marks]

Part A (6 marks):
Write SQL statements for the following:

1. **DDL (2 marks):**
   - Alter the `Books` table to add a new column `Publisher` (VARCHAR 100)
   - Create an index on the `Books` table for the `Genre` column
2. **DML (2 marks):**
   - Insert a new member: 'John Doe', 'john@email.com', '555-1234', 'Student', joined today, from 'New York'
   - Update all books published before 2000 to increase their price by 10%
3. **DCL (1 mark):**
   - Grant SELECT and INSERT permissions on the `Books` table to a user named 'Librarian'
4. **TCL (1 mark):**
   - Write a transaction that transfers a book from one member to another (close old loan, create new loan). Include COMMIT/ROLLBACK logic.

Part B (4 marks):
Explain the difference between DELETE, TRUNCATE, and DROP commands with examples. When would you use each?

Question 3: Constraints & Data Integrity [10 Marks]

Part A (6 marks):
Create a table named `Reservations` with the following constraints:

- ReservationID (Primary Key)
- MemberID (Foreign Key to Members, required)
- BookID (Foreign Key to Books, required)
- ReservationDate (defaults to current date)

- ExpiryDate (must be after ReservationDate)
- Status (must be one of: 'Active', 'Fulfilled', 'Cancelled', 'Expired')
- Priority (integer between 1 and 5)

Write the complete CREATE TABLE with all constraints including CHECK constraints.

Part B (4 marks):
For the `Loans` table, write SQL statements to:

1. Add a CHECK constraint ensuring `ReturnDate` is always after or equal to `LoanDate`
2. Add a DEFAULT constraint setting `LateFee` to 0.00
3. Explain what happens when you try to delete a Member who has active loans (referential integrity)

Question 4: String, Date, and Numeric Functions [10 Marks]

Part A (5 marks):
Write SQL queries using appropriate functions:

1. Display all member names in uppercase with their email addresses in lowercase (1 mark)
2. Extract the domain name from email addresses (everything after @) for all members (1.5 marks)
3. Calculate the age of each member based on their join date in years and months (1.5 marks)
4. Round all book prices to the nearest integer and show the difference from original price (1 mark)

Part B (5 marks):
Write a stored procedure named `CalculateLateFee` that:

- Takes parameters: @LoanID, @ReturnDate
- Calculates days overdue (difference between return date and due date)
- Calculates late fee: $0.50 per day for first 7 days, then $1.00 per day after
- Returns the calculated fee rounded to 2 decimal places
- Updates the `Loans` table with the calculated fee

Question 5: NULL Handling & Type Conversions [10 Marks]

Part A (4 marks):
Write queries to demonstrate the difference between:

1. `ISNULL` vs `COALESCE` - provide an example where COALESCE is more useful (2 marks)

2. NULLIF - write a query that prevents division by zero when calculating average price per page (1 mark)
3. Show how NULL affects calculations - demonstrate with a SUM example (1 mark)

Part B (6 marks):
Write SQL queries for the following conversions and formatting:

1. Convert today's date to these formats: (2 marks)
    - YYYY-MM-DD
    - Month DD, YYYY
    - DD/MM/YYYY
2. Write a query that safely converts the PhoneNumber to BIGINT, handling invalid data by showing 'Invalid' (2 marks)
3. Create a query that uses COLLATE to perform:
    - Case-sensitive search for books with 'SQL' in the title (exactly 'SQL', not 'sql') (1 mark)
    - Case-insensitive search for member names (1 mark)

Question 6: WHERE, Sorting, Grouping & Aggregates [10 Marks]

Part A (6 marks):
Write SQL queries for:

1. Find all programming books priced between 500 and 1000, published after 2020, sorted by price descending (1.5 marks)
2. List members from 'New York' or 'Boston' who joined in the last 6 months, sorted by join date (1.5 marks)
3. Show the top 5 most expensive books with their rank (use appropriate ranking function) (1.5 marks)
4. Find books where title contains 'Data' AND (author contains 'Smith' OR price > 700) (1.5 marks)

Part B (4 marks):
Create a comprehensive report showing:

- Genre-wise book statistics: Total books, Average price, Min price, Max price, Total inventory value
- Only include genres with more than 5 books
- Having average price > 400
- Sort by total inventory value descending

Include column aliases and appropriate formatting.

Question 7: Subqueries [10 Marks]

Part A (4 marks):
Write SQL queries using subqueries:

1. Find all members who have borrowed more books than the average number of books borrowed per member (2 marks)
2. List books that have never been borrowed (use EXISTS) (1 mark)
3. Find the member who has paid the highest total amount in fines (1 mark)

Part B (6 marks):
Using Common Table Expressions (CTE):

1. Create a CTE that calculates total loans, active loans, and overdue loans for each member (2 marks)
2. Using the above CTE, create a final query that categorizes members as:
   - 'Excellent' (no overdue, >= 5 loans)
   - 'Good' (no overdue, < 5 loans)
   - 'Warning' (1-2 overdue)
   - 'Suspended' (more than 2 overdue)

   Show member name, total loans, overdue count, and category (4 marks)

Question 8: Temporary Tables & Table Variables [10 Marks]

Part A (5 marks):
Explain the differences between Temporary Tables and Table Variables. Create a comparison table showing:

- Storage location
- Scope
- When to use each
- Performance considerations
- Transaction support

Part B (5 marks):
Write a procedure that uses a temporary table to:

1. Create a temp table `#MonthlyReport` with columns: Month, TotalLoans, UniqueMembers, TotalRevenue, OverdueCount

2. Populate it with data for the last 12 months
3. Add a calculated column showing percentage change from previous month
4. Return the final results sorted by month
5. Clean up the temp table

Question 9: Control Flow & UNION [10 Marks]

Part A (5 marks):
Write a stored procedure `ProcessOverdueLoans` that:

1. Uses a WHILE loop to process overdue loans one by one
2. For each overdue loan:
    - Calculate days overdue
    - IF days overdue > 30: Send 'Final Notice' and mark as 'Suspended'
    - ELSE IF days overdue > 14: Send 'Second Notice'
    - ELSE IF days overdue > 7: Send 'First Notice'
3. Use a counter to limit processing to 50 loans per execution
4. Include appropriate BREAK and CONTINUE logic
5. Return total loans processed

Part B (5 marks):
Create a unified contact list query using UNION that combines:

1. Active members (Name, Email, Phone, 'Member' as Type, JoinDate as AssociatedDate)
2. Staff (Name, Email, Phone, 'Staff' as Type, HireDate as AssociatedDate)
3. Suppliers (CompanyName, ContactEmail, ContactPhone, 'Supplier' as Type, ContractDate)

Requirements:

- Remove duplicates
- Sort by Type, then Name
- Show only contacts from last 2 years
- Include count by type in the result

Question 10: Joins - Comprehensive [10 Marks]

Part A (6 marks):
Write SQL queries using different join types:

1. **INNER JOIN (2 marks):** Create a detailed borrowing report showing:
    - Member name, email
    - Book title, author, genre
    - Loan date, due date, return date
    - Days borrowed
    - Only for loans from the last 3 months
2. **LEFT JOIN (2 marks):** List ALL books with their borrowing statistics:
    - Book title, author, price
    - Total times borrowed (0 if never borrowed)
    - Last borrow date (NULL if never borrowed)
    - Status: 'Popular' (>10 borrows), 'Moderate' (5-10), 'Low' (1-4), 'Never Borrowed' (0)
3. **SELF JOIN (2 marks):** Find all pairs of books by the same author where one book is more than twice the price of the other
    - Show both book titles, author name, both prices, price difference

Part B (4 marks):
Create a complex query using multiple joins:

Show a comprehensive member activity report:

- Member name, type, city
- Total books borrowed (all time)
- Currently borrowed books (count and comma-separated titles)
- Total fines (paid and unpaid separately)
- Last activity date (most recent loan or fine payment)
- Member status based on activity

Use appropriate joins (INNER, LEFT, etc.) and include members even if they have no activity.

Bonus Question (Optional - 5 Extra Marks)

Write a complex stored procedure that generates a monthly library report including:

1. New members joined this month

2. Total loans vs last month (with percentage change)
3. Top 5 most borrowed books
4. Genre-wise borrowing trends
5. Members with outstanding fines
6. Books that need reordering (borrowed more than available quantity)

Use temp tables, CTEs, joins, and control flow appropriately.

Evaluation Criteria

For each question, marks will be awarded for:

- **Correctness (50%):** Query produces expected results
- **Efficiency (20%):** Optimal use of joins, indexes, and query structure
- **Best Practices (15%):** Proper formatting, aliases, comments
- **Completeness (15%):** All requirements addressed

Good Luck! 🍀

Note to Students:

- Write readable, well-formatted SQL code
- Use meaningful aliases and column names
- Add comments for complex logic
- Test your queries mentally for edge cases
- Manage your time: ~12 minutes per question

End of Examination Paper