Report DBMS project
Online Book Store

Requirements:

Write a report which will have the following structure:
- Introduction to the system
- ER diagram
- Explanation of why the structure follows normal forms
- Explanation and coding part of each item

1. Introduction to the system

The Online Book Store is a database management system that manages information related to customers, authors, books, orders, payments, genres, and book data. This system is designed to provide an online platform for customers to buy books and manage their orders. The SQL code includes the creation of several tables such as Customer, Author, Book, Genre, and Order, and functions/procedures that help manage and manipulate data.

2. ERD:

You can see the ERD in the ***./ERD/ERD.png*** folder in the GitHub repository.

3. Explanation of why the structure follows normal forms (1NF, 2NF, 3NF):

The database structure follows the normal forms to ensure that the data is organized and free from redundancy, inconsistencies, and anomalies. The first normal form (1NF) is achieved as each column contains atomic values, and each record has a unique identifier (primary key). The second normal form (2NF) is attained as the non-key attributes are functionally dependent on the primary key. The third normal form (3NF) is also met as there is no transitive dependency between non-key attributes. For instance, the Customer table has a unique identifier 'id' and contains attributes such as 'name,' 'country,' 'city,' and 'birthdate' that are functionally dependent on the primary key. Similarly, the Book table has attributes such as 'title,' 'edition,' 'language,' and 'count_of_page' that depend on the primary key.

4. Explanation and coding part of each item:

a. tables.sql: This script creates tables for the Online Book Store system, including Customer, Author, Book, Genre, Order, BookData, BookGenre, CustomerData, Order_book, and Payment. Each table has attributes that define the unique identifier (primary key), and foreign keys to ensure data consistency and referential integrity. For instance, the Order table has a primary key 'id' and a foreign key 'customer_id' that references the Customer table.

b. count_of_data.sql: This script creates a function 'count_records' that takes a table name as input and returns the number of records in that table. The function uses dynamic SQL to execute the SELECT statement and count the number of rows. For example, SELECT count_records('Book') returns the number of books in the Book table.

c. user-defined_exception.sql: This script creates a function 'insert_book_title' that takes input parameters such as 'title,' 'edition,' 'language,' 'count_of_page,' and 'author_id' and inserts a new record in the Book table. The function raises an exception if the book title is less than five characters long. For example, SELECT insert_book_title('Abay', '3.0', 'Kazakh', 359, 1) raises an exception as the book title 'Abay' is less than five characters long, whereas SELECT insert_book_title('Abay Zholy', '3.0', 'Kazakh', 359, 1) successfully inserts a new record in the Book table.

d. determine_rows_affected.sql: This script creates two procedures 'get_books_by_author' and 'get_books_by_all_authors' that retrieve books by author and all authors, respectively. The 'get_books_by_author' procedure takes an input parameter 'p_author_name' and uses a cursor to fetch book titles from the Book table that match the author name. The 'get_books_by_all_authors' procedure uses a loop to call the 'get_books_by_author' procedure for all authors in the Author table. For instance, CALL get_books_by_all_authors() retrieves all book titles by all authors in the database.

e. groupby_information.sql: This script creates a procedure 'group_books_by_genre' that groups books by genre and displays the total count of books in each genre. The procedure uses the GROUP BY clause to group books by genre, and the COUNT(*) function to count the number of books in each group. For example, CALL group_books_by_genre() returns a list of book genres and the total count of books in each genre.

f. show_current_row_number.sql: This script creates a trigger 'show_current_row_number' that displays the current row number being inserted in the Book table. The trigger uses the ROW_COUNT() function to get the number of rows affected by the previous statement and increments a variable 'row_num' to display the current row number. For example, INSERT INTO Book (title, edition, language, count_of_page, author_id) VALUES ('The Great Gatsby', '1st', 'English', 180, 1) triggers the 'show_current_row_number' trigger, which displays the message 'Inserted row number: 1' in the console.