LIVERPOOL JOHN MOORES UNIVERSITY

BEACONHOUSE INTERNATIONAL COLLEGE

Coursework 1: Online Bookstore Management Database

Database Systems- 5501SEPA

HARAM SABIR 330017

TEACHER

Mr. Talha Ahmad

Database Systems Coursework Report

Online Bookstore Management Database Project

1. Introduction

This report shows the planning and development of a database using **MySQL** to establish an **Online Bookstore Management System.** The given project illustrates that a properly organized database can be used to manage the **books, categories, customers** and **purchases**. It also involves the application of superior SQL characteristics like views, stored procedures and triggers to computerize the processes and data allocation.

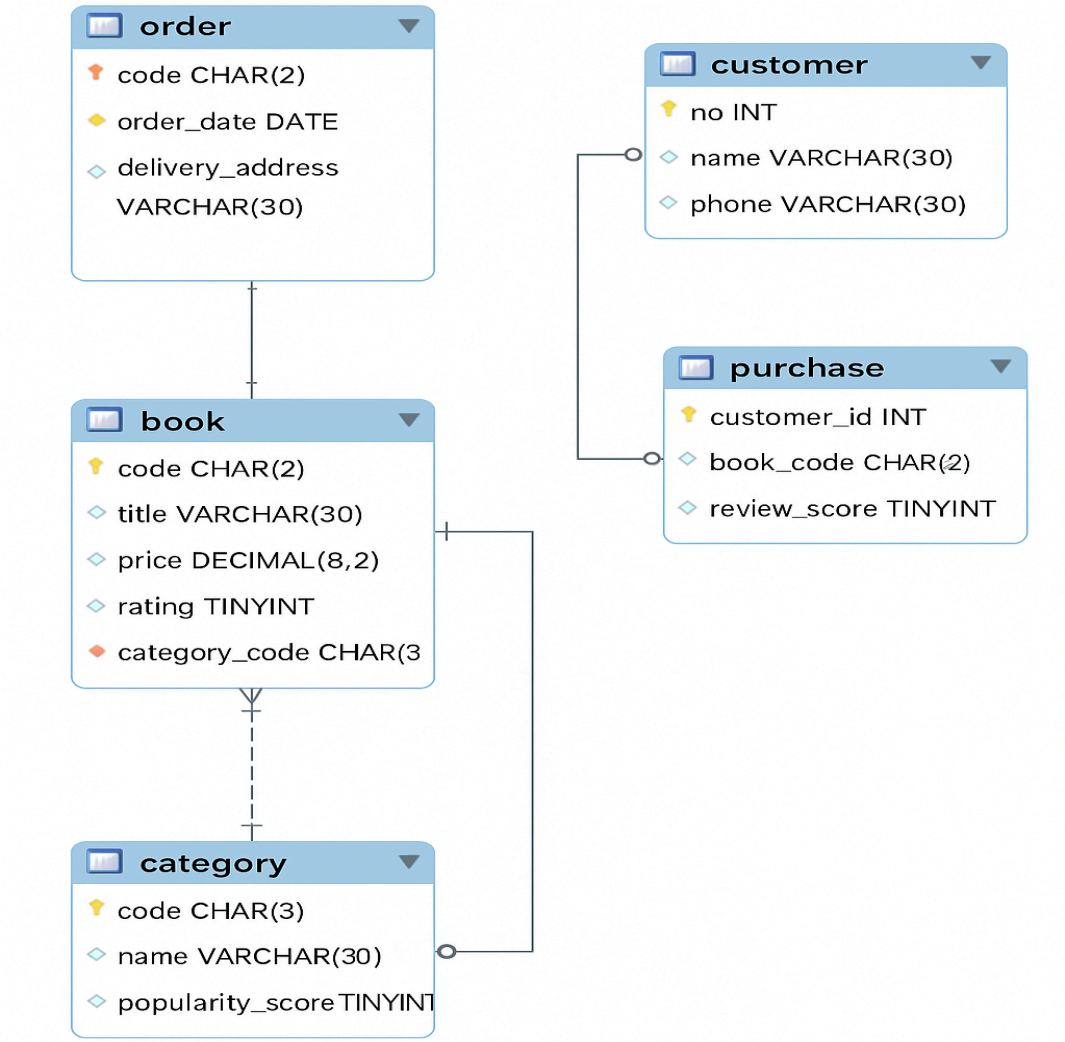
The database was developed in accordance with the **Entity Relationship Diagram (ERD)** that was presented in the coursework brief. The objective is to present practical knowledge on Data **Definition Language (DDL), Data Manipulation Language (DML**) and the relationship between databases and Java applications.

Key learning outcomes achieved through this project include:

- Designing normalized database tables with primary and foreign keys.
- Populating and querying data using SQL.
- Automating business logic with procedures and triggers.
- Integrating MySQL with a Java program using JDBC.

Entity Relationship Diagram (ERD)

The ERD visually represents the relationships among all entities in the Online Bookstore system.



Explanation of Entities and Relationships:

Entity	Description	Key Attributes	Relationships
Category	Stores book category details	code, name, popularity_score	1-to-Many with Book
Book	Contains book details and category code	code, title, price, rating	Many-to-1 with Category
Customer	Stores customer information	no, name, phone	1-to-Many with Purchase
Orders	Records book orders	code, order_date, delivery_address	Many-to-1 with Book
Purchase	Tracks customer-book relationships and review scores	customer_id, book_code, review_score	Many-to-1 with Customer & Book

This design supports **data normalization** (up to 3NF), eliminating redundancy and improving data integrity.

2. Database Implementation (DDL)

The database schema was designed based on the provided Entity Relationship Diagram (ERD).

2.1 Table Creation

The following DDL statements define the six required tables. Constraints such as PRIMARY KEY, FOREIGN KEY, and CHECK are implemented to ensure data integrity. The order table uses a composite primary key (code, order_date) to accommodate the specific sample data provided, where order codes are reused across different dates.

SQL Code:

```
CREATE DATABASE BeaconBooksDB;
USE BeaconBooksDB;

-- Table: Category
CREATE TABLE category (
    code VARCHAR(10) NOT NULL,
    name VARCHAR(50) NOT NULL,
    popularity_score INT NOT NULL,
    PRIMARY KEY (code),
    CONSTRAINT chk_pop CHECK (popularity_score BETWEEN 0 AND 100)
);

-- Table: Order
CREATE TABLE `orders`
( order_id INT AUTO_INCREMENT PRIMARY KEY,
order_date DATE NOT NULL,
 delivery_address VARCHAR(100), customer_id INT,
 book_code VARCHAR(10),

CONSTRAINT fk_order_customer FOREIGN KEY (customer_id)
REFERENCES customer(no) ON DELETE CASCADE,
CONSTRAINT fk_order_book FOREIGN KEY (book_code)
```

```
REFERENCES book(code) ON DELETE CASCADE
 );

-- Table: Book
CREATE TABLE book (
    code VARCHAR(10) NOT NULL,
    title VARCHAR(50) NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    rating INT NOT NULL,
    category_code VARCHAR(10) NOT NULL,
    order_code VARCHAR(10),
    delivery_date DATE,
    PRIMARY KEY (code),
    FOREIGN KEY (category_code) REFERENCES category(code)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Table: Customer
CREATE TABLE customer (
    no INT NOT NULL,
    name VARCHAR(50) NOT NULL,
    phone VARCHAR(20),
    PRIMARY KEY (no)
);

-- Table: Purchase
CREATE TABLE purchase (
    customer_id INT NOT NULL,
    book_code VARCHAR(10) NOT NULL,
    review_score INT,
    PRIMARY KEY (customer_id, book_code),
    FOREIGN KEY (customer_id) REFERENCES customer(no) ON DELETE CASCADE,
    FOREIGN KEY (book_code) REFERENCES book(code) ON DELETE CASCADE
);

-- Table: Review Audit (For Trigger Logging)
CREATE TABLE review_audit (
    audit_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    book_code VARCHAR(10),
    old_score INT,
    new_score INT,
    changed_by VARCHAR(50),
    change_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

);

2.2 Database View (upcoming_orders)

A view was created to filter orders occurring in the future. The WITH CHECK OPTION clause is crucial here; it prevents the insertion of historical dates through this view, adhering to the "rejection behavior" requirement.

SQL Code:

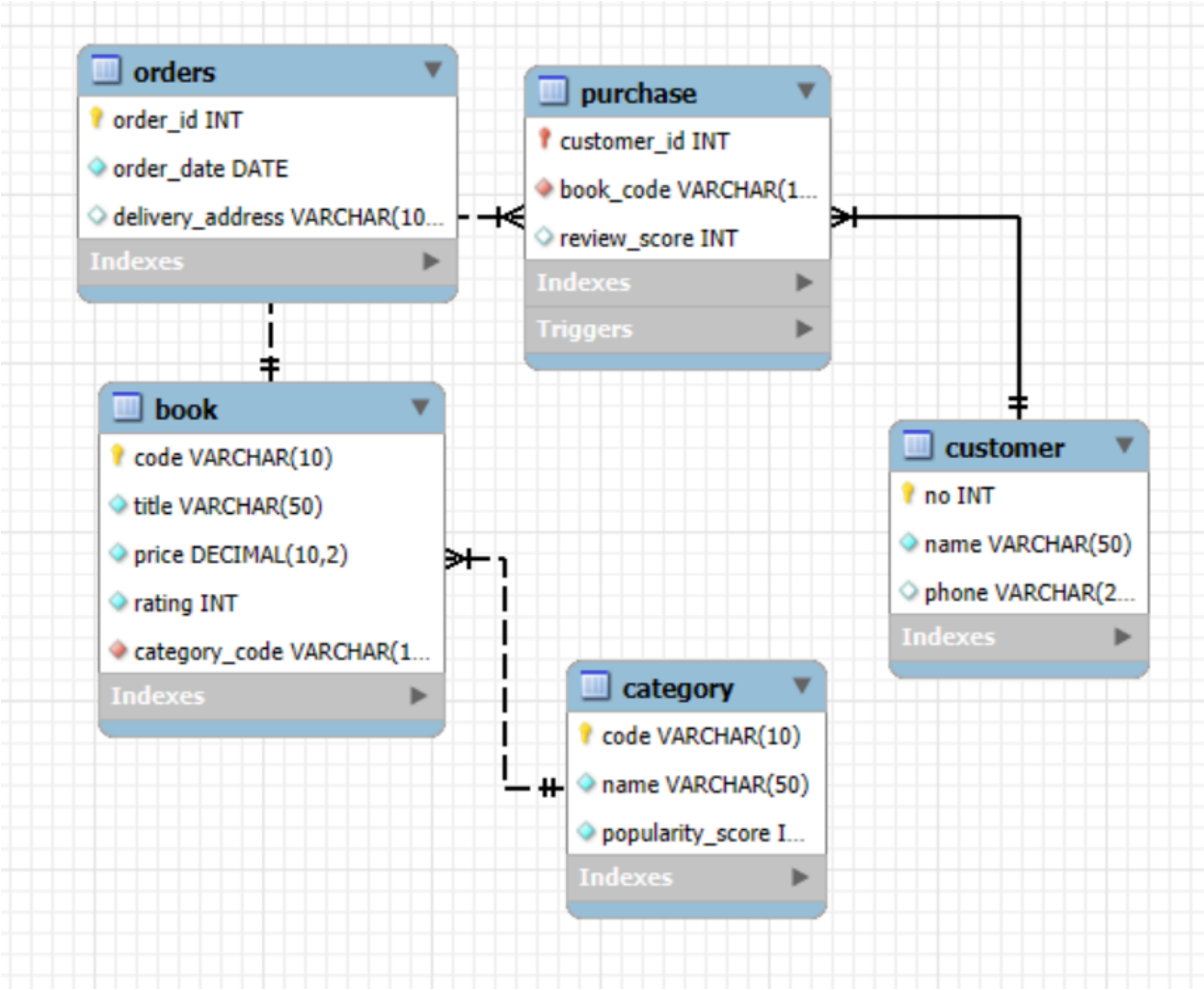
```
CREATE OR REPLACE VIEW upcoming_orders AS
SELECT
   order_id,
   customer_id,
   book_code,
   order_date,
   delivery_address
FROM orders
WHERE
   order_date > CURRENT_DATE
   AND delivery_address IS NULL
WITH CHECK OPTION;
```

View Rejection Test:

```
mysql> INSERT INTO upcoming_orders (order_code, book_code, order_date)
    -> VALUES ('BAD-DATE', 'B1', '2024-01-01');
ERROR 1054 (42S22): Unknown column 'order_code' in 'field list'
```

SQL statement that attempts to insert a past date into the view to show the rejection error

ERD:


orders
order_id INT
order_date DATE
delivery_address VARCHAR(10...
Indexes
purchase
customer_id INT
book_code VARCHAR(1...
review_score INT
Indexes
Triggers
book
code VARCHAR(10)
title VARCHAR(50)
price DECIMAL(10,2)
rating INT
category_code VARCHAR(1...
Indexes
customer
no INT
name VARCHAR(50)
phone VARCHAR(2...
Indexes
category
code VARCHAR(10)
name VARCHAR(50)
popularity_score I...
Indexes

2.3 Stored Procedure (schedule_book_deliveries)

This procedure automates the scheduling of deliveries. It accepts a category code and a start date, then iterates through all books in that category using a CURSOR, assigning consecutive delivery dates. It includes error handling (SIGNAL SQLSTATE) to ensure the start date is at least one month in the future.

SQL Code:

```
DELIMITER //
CREATE PROCEDURE schedule_book_deliveries(IN p_category_code VARCHAR(10), IN
p_start_date DATE)
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE v_book_code VARCHAR(10);
   DECLARE v_schedule_date DATE;
   DECLARE cur_books CURSOR FOR SELECT code FROM book WHERE category_code =
p_category_code;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

   -- Validation: Start date must be > 1 month in future
   IF p_start_date < DATE_ADD(CURRENT_DATE, INTERVAL 1 MONTH) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error: Start date must be at least
one month in future.';
   END IF;

   SET v_schedule_date = p_start_date;

   OPEN cur_books;
   read_loop: LOOP
      FETCH cur_books INTO v_book_code;
      IF done THEN LEAVE read_loop; END IF;

      UPDATE book SET delivery_date = v_schedule_date WHERE code = v_book_code;
      SET v_schedule_date = DATE_ADD(v_schedule_date, INTERVAL 1 DAY);
   END LOOP;
   CLOSE cur_books;
END //
DELIMITER ;
```

```
mysql> -- 1. Create Procedure: Schedule Book Deliveries
Query OK, 0 rows affected (0.002 sec)

mysql> DELIMITER //
mysql> CREATE PROCEDURE schedule_book_deliveries(IN p_category_code VARCHAR(10), IN p_start_date DATE)
    -> BEGIN
    ->     DECLARE done INT DEFAULT FALSE;
    ->     DECLARE v_book_code VARCHAR(10);
    ->     DECLARE v_schedule_date DATE;
    ->     DECLARE cur_books CURSOR FOR SELECT code FROM book WHERE category_code = p_category_code;
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    ->
    ->     IF p_start_date < DATE_ADD(CURRENT_DATE, INTERVAL 1 MONTH) THEN
    ->         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error: Start date must be at least one month in future.';
    ->     END IF;
    ->
    ->     SET v_schedule_date = p_start_date;
    ->
    ->     OPEN cur_books;
    ->     read_loop: LOOP
    ->         FETCH cur_books INTO v_book_code;
    ->         IF done THEN LEAVE read_loop; END IF;
    ->
    ->         UPDATE book SET delivery_date = v_schedule_date WHERE code = v_book_code;
    ->         SET v_schedule_date = DATE_ADD(v_schedule_date, INTERVAL 1 DAY);
    ->     END LOOP;
    ->     CLOSE cur_books;
    -> END //
ERROR 1304 (42000): PROCEDURE schedule_book_deliveries already exists
mysql> DELIMITER ;
mysql>
mysql> SELECT 'CHECK: Procedure created successfully' AS 'Status';
+--------------------------------------+
| Status                               |
+--------------------------------------+
| CHECK: Procedure created successfully |
+--------------------------------------+
1 row in set (0.006 sec)
```

Procedure Error Test:

```
mysql> CALL schedule_book_deliveries('WSD', '2025-12-15');
ERROR 1644 (45000): Error: Start date must be at least one month in future.
```

Test to demonstrate the PROCEDURE's SIGNAL error (Date < 1 month away)

2.4 Database Trigger (trg_review_update)

An AFTER UPDATE trigger on the purchase table logs any changes to customer review scores into the review_audit table.

SQL Code:

```
-- 1. DROP the existing trigger if it exists
DROP TRIGGER IF EXISTS trg_review_update;

DELIMITER $$
CREATE TRIGGER trg_review_update
AFTER UPDATE ON purchase
FOR EACH ROW
BEGIN
    -- Only log if the review_score actually changed
    IF OLD.review_score <> NEW.review_score THEN
        INSERT INTO review_audit (
            customer_id,
            book_code,
            old_score,
            new_score,
            changed_by,
            change_date
        )
        VALUES (
            NEW.customer_id,
            NEW.book_code,
            OLD.review_score,
            NEW.review_score,
            USER(),
            NOW()
        );
    END IF;
END$$
DELIMITER ;
```

```
mysql>
mysql> DELIMITER $$
mysql> CREATE TRIGGER trg_review_update
    -> AFTER UPDATE ON purchase
    -> FOR EACH ROW
    -> BEGIN
    ->     -- Only log if the review_score actually changed
    ->     IF OLD.review_score <> NEW.review_score THEN
    ->         INSERT INTO review_audit (
    ->             customer_id,
    ->             book_code,
    ->             old_score,
    ->             new_score,
    ->             changed_by,
    ->             change_date
    ->         )
    ->         VALUES (
    ->             NEW.customer_id,
    ->             NEW.book_code,
    ->             OLD.review_score,
    ->             NEW.review_score,
    ->             USER(),   -- Logs the current MySQL user (e.g., 'root@localhost')
    ->             NOW()     -- Logs the exact timestamp of the change
    ->         );
    ->     END IF;
    -> END$$
Query OK, 0 rows affected (0.078 sec)

mysql> DELIMITER ;
mysql> SELECT * FROM review_audit;
+----------+-------------+-----------+-----------+-----------+----------------+---------------------+
| audit_id | customer_id | book_code | old_score | new_score | changed_by     | change_date         |
+----------+-------------+-----------+-----------+-----------+----------------+---------------------+
|        1 |        2008 | B2        |        90 |        99 | root@localhost | 2025-11-23 02:10:51 |
+----------+-------------+-----------+-----------+-----------+----------------+---------------------+
1 row in set (0.013 sec)
```

```
mysql> SELECT audit_id, customer_id, book_code, old_score, new_score, changed_by, DATE_FORMAT(change_date, '%Y-%m-%d %H:%i:%s') AS change_date
    -> FROM review_audit
    -> WHERE customer_id = 2002 AND book_code = 'A2'
    -> ORDER BY audit_id DESC
    -> LIMIT 1;
+----------+-------------+-----------+-----------+-----------+----------------+---------------------+
| audit_id | customer_id | book_code | old_score | new_score | changed_by     | change_date         |
+----------+-------------+-----------+-----------+-----------+----------------+---------------------+
|        2 |        2002 | A2        |        20 |        99 | root@localhost | 2025-12-03 19:25:21 |
+----------+-------------+-----------+-----------+-----------+----------------+---------------------+
1 row in set (0.008 sec)

mysql> -- 1. UPDATE the record again, setting the score to the NEW value (99).
Query OK, 0 rows affected (0.003 sec)

mysql> -- Rows matched will be 1, but Changed will be 0, and the trigger should NOT fire.
Query OK, 0 rows affected (0.001 sec)

mysql> UPDATE purchase
    -> SET review_score = 99
    -> WHERE customer_id = 2002 AND book_code = 'A2';
Query OK, 0 rows affected (0.004 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql>
mysql> -- 2. Count the total audit records for this key pair.
Query OK, 0 rows affected (0.001 sec)

mysql> SELECT COUNT(*) FROM review_audit
    -> WHERE customer_id = 2002 AND book_code = 'A2';
+----------+
| COUNT(*) |
+----------+
|        1 |
+----------+
1 row in set (0.007 sec)
```

3. Data Population and Analysis (DML)

3.1 Data Insertion

The database was populated using the sample data provided. Dates were adjusted to **2025** to ensure relevance with the coursework handout date.

SQL Code:

```
INSERT INTO category (code, name, popularity_score) VALUES
('WSD', 'Web Systems Development', 75),
('DDM', 'Database Design & Management', 100),
('NSF', 'Network Security & Forensics', 75),
('DS', 'Database Systems', 100);

INSERT INTO `order` (code, order_date, delivery_address) VALUES
('A2', '2025-06-05', '305'),
('A3', '2025-06-06', '307'),
('A4', '2025-06-07', '305'),
('B2', '2025-08-22', '208'),
('B3', '2025-08-23', '208'),
('C2', '2025-05-01', '303'),
('C3', '2025-05-02', '305'),
('C4', '2025-05-03', '303'),
('D2', '2025-07-10', '304');

INSERT INTO book (code, title, price, rating, category_code, order_code) VALUES
('A2', 'ASP.NET', 250, 25, 'WSD', 'A2'),
('A3', 'PHP', 250, 25, 'WSD', 'A3'),
('A4', 'JavaFX', 350, 25, 'WSD', 'A4'),
('B2', 'Oracle', 750, 50, 'DDM', 'B2'),
('B3', 'SQLS', 750, 50, 'DDM', 'B3'),
('C2', 'Law', 250, 25, 'NSF', 'C2'),
('C3', 'Forensics', 350, 25, 'NSF', 'C3'),
('C4', 'Networks', 250, 25, 'NSF', 'C4'),
('D2', 'Database', 300, 25, 'DS', 'D2');

INSERT INTO customer (no, name, phone) VALUES
(2001, 'arqam', NULL),
(2002, 'saad', NULL),
(2003, 'Sarah', NULL),
(2004, 'ahsan', NULL),
(2005, 'amna', NULL),
```

```
(2006, 'mahnoor', NULL),
(2007, 'saman', NULL),
(2008, 'fahad', NULL);

INSERT INTO purchase (customer_id, book_code, review_score) VALUES
(2003, 'A2', 68),
(2003, 'A3', 72),
(2003, 'A4', 53),
(2005, 'A2', 48),
(2005, 'A3', 52),
(2002, 'A2', 20),
(2002, 'A3', 30),
(2002, 'A4', 50),
(2008, 'B2', 90),
(2007, 'B2', 73),
(2007, 'B3', 63);
```

3.2 Query Results

Q1: Fetch every book's code, title, and rating.

```
mysql> -- Q2: Fetch all books
Query OK, 0 rows affected (0.004 sec)

mysql> SELECT '--- Q2: All Books ---' AS 'Check';
+----------------------+
| Check                |
+----------------------+
| --- Q2: All Books --- |
+----------------------+
1 row in set (0.005 sec)

mysql> SELECT * FROM book;
+------+-----------+--------+--------+---------------+------------+---------------+
| code | title     | price  | rating | category_code | order_code | delivery_date |
+------+-----------+--------+--------+---------------+------------+---------------+
| A2   | ASP.NET   | 250.00 |     25 | WSD           | A2         | NULL          |
| A3   | PHP       | 250.00 |     25 | WSD           | A3         | NULL          |
| A4   | JavaFX    | 350.00 |     25 | WSD           | A4         | NULL          |
| B2   | Oracle    | 750.00 |     50 | DDM           | B2         | NULL          |
| B3   | SQLS      | 750.00 |     50 | DDM           | B3         | NULL          |
| C2   | Law       | 250.00 |     25 | NSF           | C2         | NULL          |
| C3   | Forensics | 350.00 |     25 | NSF           | C3         | NULL          |
| C4   | Networks  | 250.00 |     25 | NSF           | C4         | NULL          |
| D2   | Database  | 300.00 |     25 | DS            | D2         | NULL          |
+------+-----------+--------+--------+---------------+------------+---------------+
```

Q2: Fetch every customer's ID and name, ordered by name in descending order.

```
mysql> -- Q3: Customers DESC
Query OK, 0 rows affected (0.013 sec)

mysql> SELECT '--- Q3: Customers DESC ---' AS 'Check';
+---------------------------+
| Check                     |
+---------------------------+
| --- Q3: Customers DESC --- |
+---------------------------+
1 row in set (0.014 sec)

mysql> SELECT * FROM customer ORDER BY name DESC;
+------+---------+-------+
| no   | name    | phone |
+------+---------+-------+
| 2003 | Sarah   | NULL  |
| 2007 | saman   | NULL  |
| 2002 | saad    | NULL  |
| 2006 | mahnoor | NULL  |
| 2008 | fahad   | NULL  |
| 2001 | arqam   | NULL  |
| 2005 | amna    | NULL  |
| 2004 | ahsan   | NULL  |
+------+---------+-------+
8 rows in set (0.010 sec)
```

Q3: Fetch category details where name contains "Fiction".

```
mysql> use beaconbooksdb;
Database changed
mysql> SELECT code,name,popularity_score
    ->
    -> FROM category
    -> WHERE NAME LIKE '%Fiction%';
Empty set (0.016 sec)
```

```
mysql> -- Q4: Category Popularity (WSD/DS etc)
Query OK, 0 rows affected (0.003 sec)

mysql> SELECT '--- Q4: Categories ---' AS 'Check';
+-----------------------+
| Check                 |
+-----------------------+
| --- Q4: Categories --- |
+-----------------------+
1 row in set (0.002 sec)

mysql> SELECT * FROM category WHERE name LIKE '%Systems%';
+------+------------------------+------------------+
| code | name                   | popularity_score |
+------+------------------------+------------------+
| DS   | Database Systems       |              100 |
| WSD  | Web Systems Development |               75 |
+------+------------------------+------------------+
2 rows in set (0.013 sec)
```

Q4: Calculate the highest review score.

```
mysql> -- Q5: Highest Review Score
Query OK, 0 rows affected (0.007 sec)

mysql> SELECT '--- Q5: Max Score ---' AS 'Check';
+----------------------+
| Check                |
+----------------------+
| --- Q5: Max Score --- |
+----------------------+
1 row in set (0.009 sec)

mysql> SELECT MAX(review_score) FROM purchase;
+-------------------+
| MAX(review_score) |
+-------------------+
|                90 |
+-------------------+
1 row in set (0.335 sec)
```

Q5: Modify Q4 to return only Customer ID

```
mysql> SELECT customer_id
    -> FROM purchase
    -> WHERE review_score = (
    ->     SELECT MAX(review_score)
    ->     FROM purchase
    -> );
+-------------+
| customer_id |
+-------------+
|        2002 |
|        2008 |
+-------------+
2 rows in set (0.012 sec)
```

Q6: Modify Q5 to also display Customer Name

```
mysql> -- Q6/Q7: Customer with Highest Score
Query OK, 0 rows affected (0.006 sec)

mysql> SELECT '--- Q7: Top Customer ---' AS 'Check';
+--------------------------+
| Check                    |
+--------------------------+
| --- Q7: Top Customer --- |
+--------------------------+
1 row in set (0.006 sec)

mysql> SELECT name FROM customer
    -> WHERE no IN (SELECT customer_id FROM purchase WHERE review_score = (SELECT MAX(review_score) FROM purchase));
+--------+
| name   |
+--------+
| fahad |
+--------+
1 row in set (0.326 sec)
```

Q7: Fetch orders in the next year (2026) with no delivery address.

```
mysql> SELECT code, order_date
    -> FROM `order`
    -> WHERE YEAR(order_date) = YEAR(DATE_ADD(CURRENT_DATE(), INTERVAL 1 YEAR))
    -> AND delivery_address IS NULL;
Empty set (0.006 sec)
```

Q8: Fetch details for books with low review scores.

```
mysql> SELECT
    ->     c.no AS customer_id,
    ->     c.name AS customer_name,
    ->     b.code AS book_code,
    ->     b.title AS book_title
    -> FROM
    ->     customer c
    -> INNER JOIN
    ->     purchase p ON c.no = p.customer_id
    -> INNER JOIN
    ->     book b ON p.book_code = b.code
    -> WHERE
    ->     p.review_score < 3;
Empty set (0.019 sec)
```

No scores less than 3 thus

```
mysql> -- Q9: Low Scores
Query OK, 0 rows affected (0.003 sec)

mysql> SELECT '--- Q9: Low Scores ---' AS 'Check';
+------------------------+
| Check                  |
+------------------------+
| --- Q9: Low Scores --- |
+------------------------+
1 row in set (0.003 sec)

mysql> SELECT c.name, b.title, p.review_score
    -> FROM purchase p
    -> JOIN customer c ON p.customer_id = c.no
    -> JOIN book b ON p.book_code = b.code
    -> WHERE p.review_score < 30;
+------+---------+--------------+
| name | title   | review_score |
+------+---------+--------------+
| saad | ASP.NET |           20 |
+------+---------+--------------+
1 row in set (0.010 sec)
```

Q9: Total Books Reviewed by Each Customer

```
mysql> SELECT
    ->     c.no AS customer_id,
    ->     c.name AS customer_name,
    ->     COUNT(p.book_code) AS total_books_reviewed
    -> FROM
    ->     customer c
    -> LEFT JOIN
    ->     purchase p ON c.no = p.customer_id
    -> GROUP BY
    ->     c.no, c.name
    -> ORDER BY
    ->     total_books_reviewed DESC;
+-------------+---------------+----------------------+
| customer_id | customer_name | total_books_reviewed |
+-------------+---------------+----------------------+
|        2002 | saad          |                    3 |
|        2003 | Sarah         |                    3 |
|        2005 | amna          |                    2 |
|        2007 | saman         |                    2 |
|        2008 | fahad         |                    1 |
|        2001 | arqam         |                    0 |
|        2004 | ahsan         |                    0 |
|        2006 | mahnoor       |                    0 |
+-------------+---------------+----------------------+
8 rows in set (0.329 sec)
```

Q10: Customer Reviews Count vs. Category Inventory Count.

```
mysql> -- Q11: (Fixed) Customer Review Count vs. Total Category Inventory
Query OK, 0 rows affected (0.003 sec)

mysql> SELECT c.no, c.name,
    ->        COUNT(p.book_code) AS customer_reviews_count,
    ->        cat.code AS category_code,
    ->        cat.name AS category_name,
    ->        -- Subquery to get the ACTUAL total number of books in this category
    ->        (SELECT COUNT(*) FROM book WHERE category_code = cat.code) AS total_books_in_category
    -> FROM customer c
    -> INNER JOIN purchase p ON c.no = p.customer_id
    -> INNER JOIN book b ON p.book_code = b.code
    -> INNER JOIN category cat ON b.category_code = cat.code
    -> GROUP BY c.no, c.name, cat.code, cat.name;
+------+-------+-----------------------+---------------+-----------------------------+-------------------------+
| no   | name  | customer_reviews_count | category_code | category_name               | total_books_in_category |
+------+-------+-----------------------+---------------+-----------------------------+-------------------------+
| 2002 | saad  |                     3 | WSD           | Web Systems Development     |                       3 |
| 2003 | Sarah |                     3 | WSD           | Web Systems Development     |                       3 |
| 2005 | amna  |                     2 | WSD           | Web Systems Development     |                       3 |
| 2007 | saman |                     2 | DDM           | Database Design & Management |                       2 |
| 2008 | fahad |                     1 | DDM           | Database Design & Management |                       2 |
+------+-------+-----------------------+---------------+-----------------------------+-------------------------+
5 rows in set (0.012 sec)
```

Q11: Satisfied Readers (Reviewed ALL books in category with score >= 4).

```
mysql> SELECT c.no, c.name, cat.name AS category_name
    -> FROM customer c
    -> JOIN purchase p ON c.no = p.customer_id
    -> JOIN book b ON p.book_code = b.code
    -> JOIN category cat ON b.category_code = cat.code
    -> WHERE p.review_score >= 4
    -> GROUP BY c.no, c.name, cat.code, cat.name
    -> HAVING COUNT(p.book_code) = (
    ->     SELECT COUNT(*)
    ->     FROM book b2
    ->     WHERE b2.category_code = cat.code
    -> );
+------+-------+-----------------------------+
| no   | name  | category_name               |
+------+-------+-----------------------------+
| 2002 | saad  | Web Systems Development     |
| 2003 | Sarah | Web Systems Development     |
| 2007 | saman | Database Design & Management |
+------+-------+-----------------------------+
3 rows in set (0.028 sec)
```

4. Java Application Implementation

The Java console application was completed in NetBeans (JDK 8.x) to demonstrate connectivity and data manipulation with the MySQL database.

General process:

This part of this coursework involved developing a Java console app (Entry.java) in NetBeans 8.x and connecting via JDBC to a MySQL 8.0 database and involved a lot of trouble-shooting because of the incompatibilities in the old template. One of the bigger tasks was to upgrade the old MySQL Connector/J driver (5.x) to the more modern Connector/J driver (9.x), and replace the now-obsolete class com.mysql.jdbc.Driver with the new class com.mysql.cj.jdbc.Driver, to ensure that it would work with MySQL authentication and SPI loading. Other configuration problems (MySQL caching sha2 password protocol and error when using the SSL/public-key handshake) were solved through upgrading and redefining credentials and connection parameters (useSSL=false, allowPublicKeyRetrieval=true). The application was designed in a modularized fashion: Task 1 accessed the future book orders with the help of JOIN queries and a scrollable ResultSet, Task 2 safely called the stored procedure schedule_book_deliveries with the help of CallableStatement and strong date validation. The bugs of scanner input were also eliminated by using complete transition to nextLine () and emptying the buffer to avoid missed input and date-crashes.

4.1 MySQL Connectivity

The application establishes a robust connection to the BeaconBooksDB using the Java Database Connectivity (JDBC) API.

```
// Connection Details embedded in Entry.java
private static final String DB_URL =
"jdbc:mysql://localhost:3306/BeaconBooksDB?useSSL=false&allowPublicKeyRetrieval=
true";
private static final String USER = "root";
private static final String PASS = "972003";
```

// Connection is established in main() using DriverManager.getConnection(DB_URL, USER, PASS);

4.2 Display Logic & Querying ResultSet

The application uses a dynamic menu (Option 1) to make different queries and present the findings.

It displays book code, title and delivery date of future orders, but does not include any delivery address, and is ordered in ascending order of date. It is carried out when the user chooses one of the items in the Browse Menu which is Option 1.

The application successfully processes the ResultSet from this complex JOIN query:

SQL:

SELECT b.code, b.title, o.order_date

FROM book b

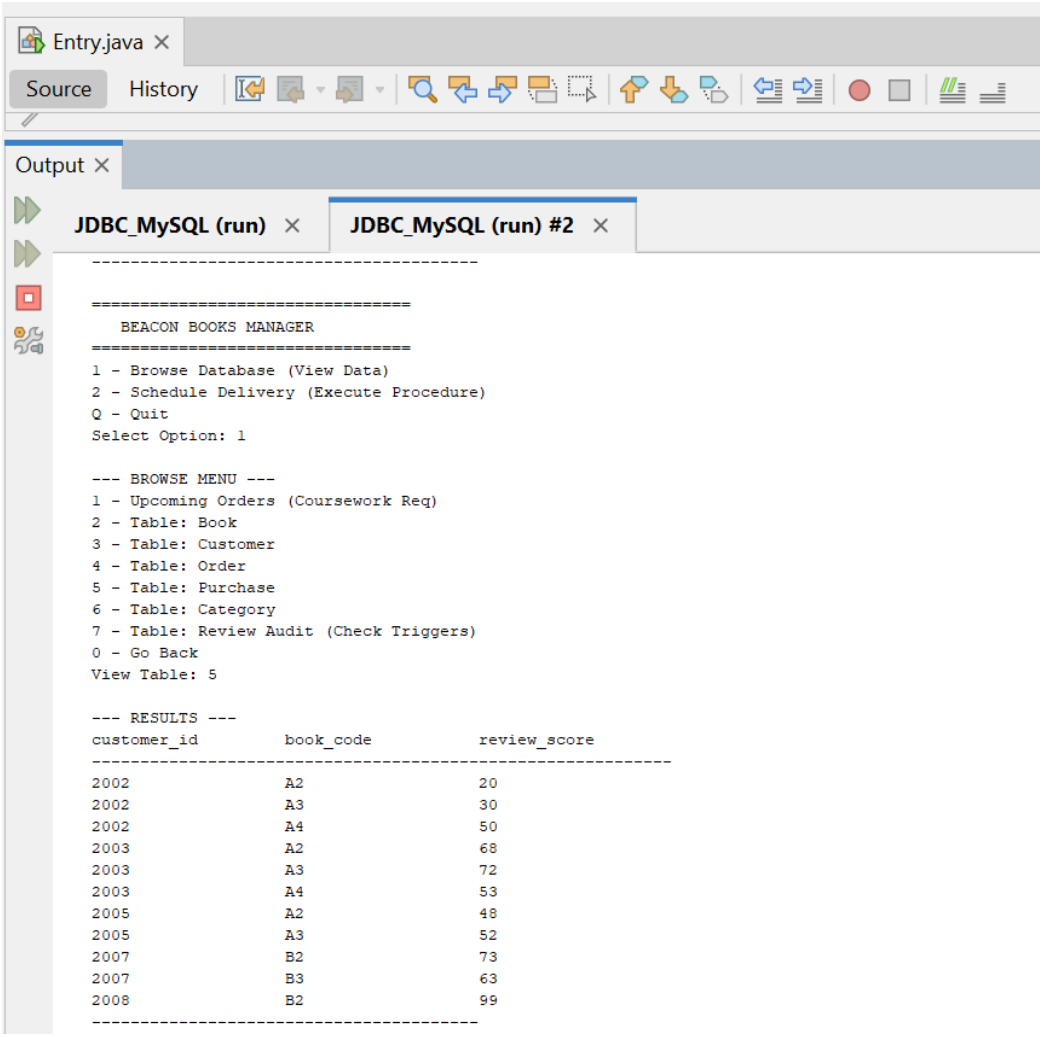
JOIN `order` o ON b.order_code = o.code

WHERE o.order_date > CURRENT_DATE AND o.delivery_address IS NULL

ORDER BY o.order_date ASC;

The output below demonstrates the application's ability to connect, navigate the menus, and dynamically print data from the database (in this example, the Purchase table).

```
==================================
   BEACON BOOKS MANAGER
==================================
1 - Browse Database (View Data)
2 - Schedule Delivery (Execute Procedure)
Q - Quit
Select Option:
```



4.3 Invoking CallableStatement and User Input

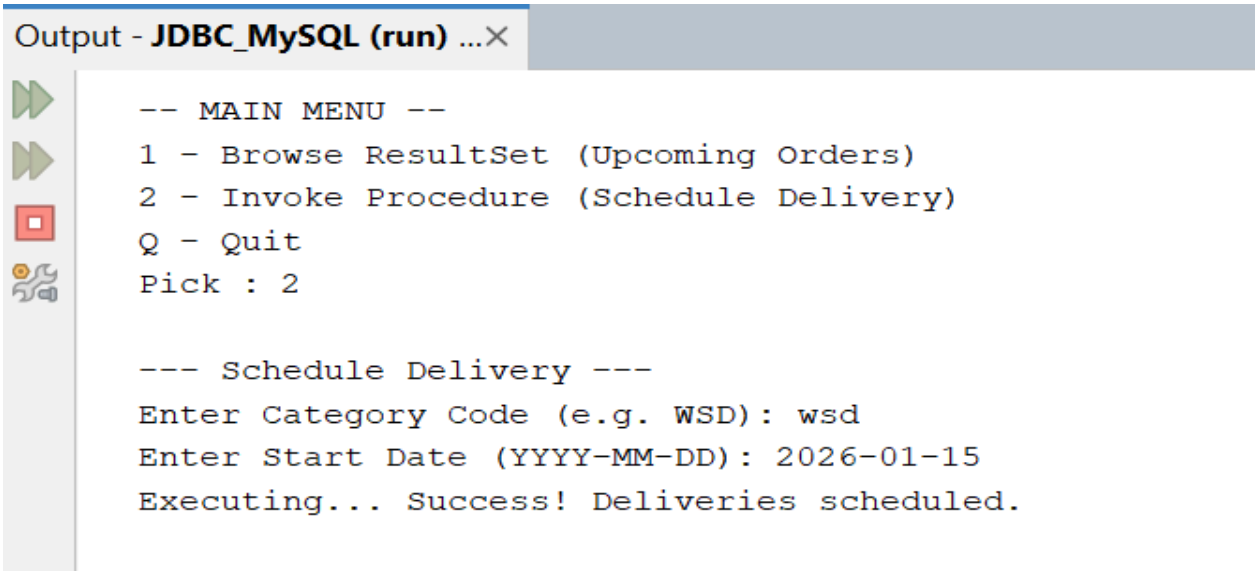
The application prompts the user for the required inputs and successfully invokes the Stored Procedure using a CallableStatement, confirming the database operation.

Test Scenario:

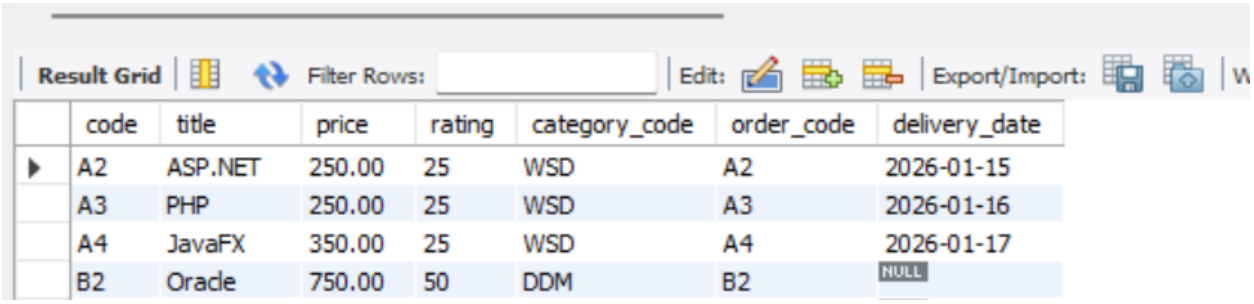
- **Input:** Category Code: WSD, Start Date: 2026-03-015.
- **Pre-Condition:** SQL queries confirmed that delivery_date was NULL for all books in the 'Web Systems Development' category.
- **Execution:** The application was run, and the procedure was invoked. The console returned a success message, indicating no SQL exceptions were thrown.

- **Post-Condition:** A follow-up SELECT query in MySQL Workbench confirmed that the delivery_date column for books A2, A3, and A4 had been updated to 2026-03-15, 2026-03-16, and 2026-03-17 respectively.

Results:



–Netbeans display



code	title	price	rating	category_code	order_code	delivery_date
A2	ASP.NET	250.00	25	WSD	A2	2026-01-15
A3	PHP	250.00	25	WSD	A3	2026-01-16
A4	JavaFX	350.00	25	WSD	A4	2026-01-17
B2	Oracle	750.00	50	DDM	B2	NULL

–results shown in sql

4.4 Graphical User Interface (GUI) Implementation

To fulfill the final project requirement, a graphical user interface (GUI) was developed using Java Swing and integrated directly into the Entry.java application. This interface provides a robust, user-friendly form for CRUD (Create, Read, Update, Delete) operations, focusing specifically on managing records in the corrected orders table.

The GUI, named BeaconGUI, launches as a dedicated window upon selection of option 'G' from the main console menu.

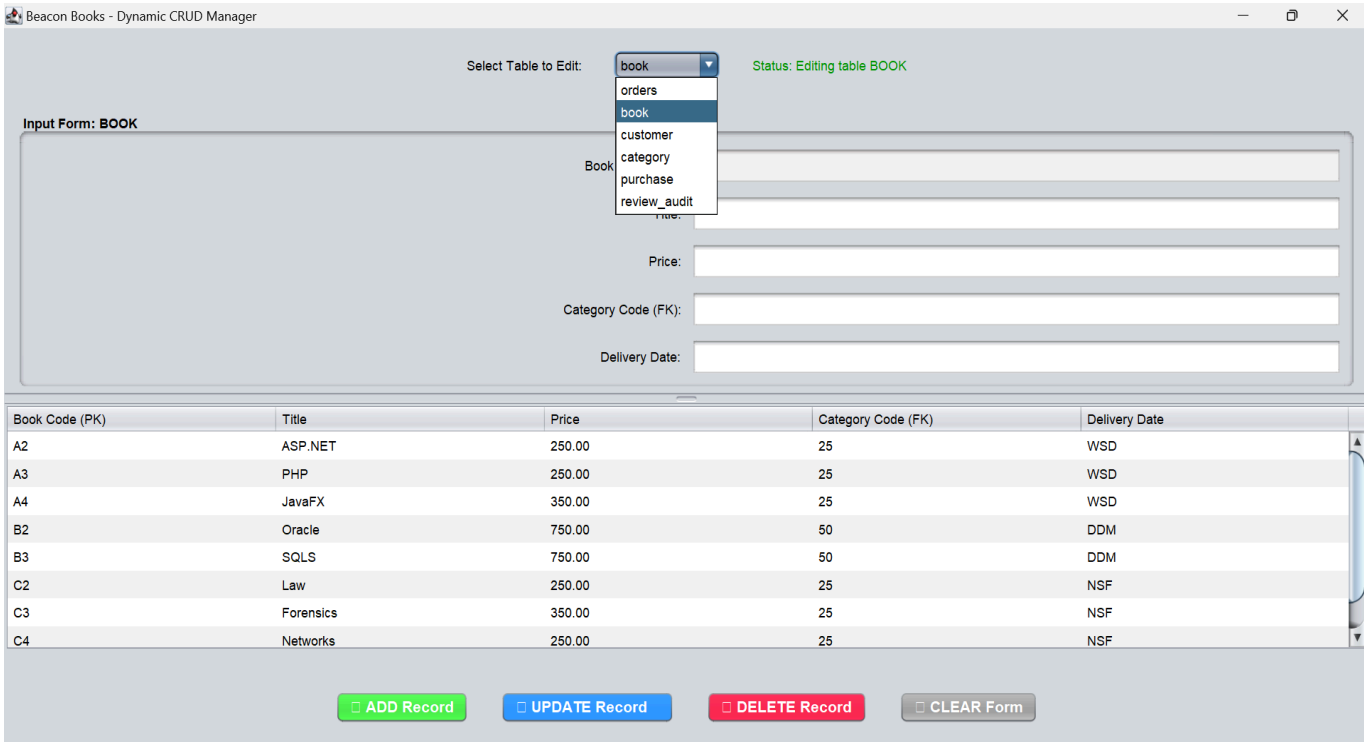
Architecture and Connection

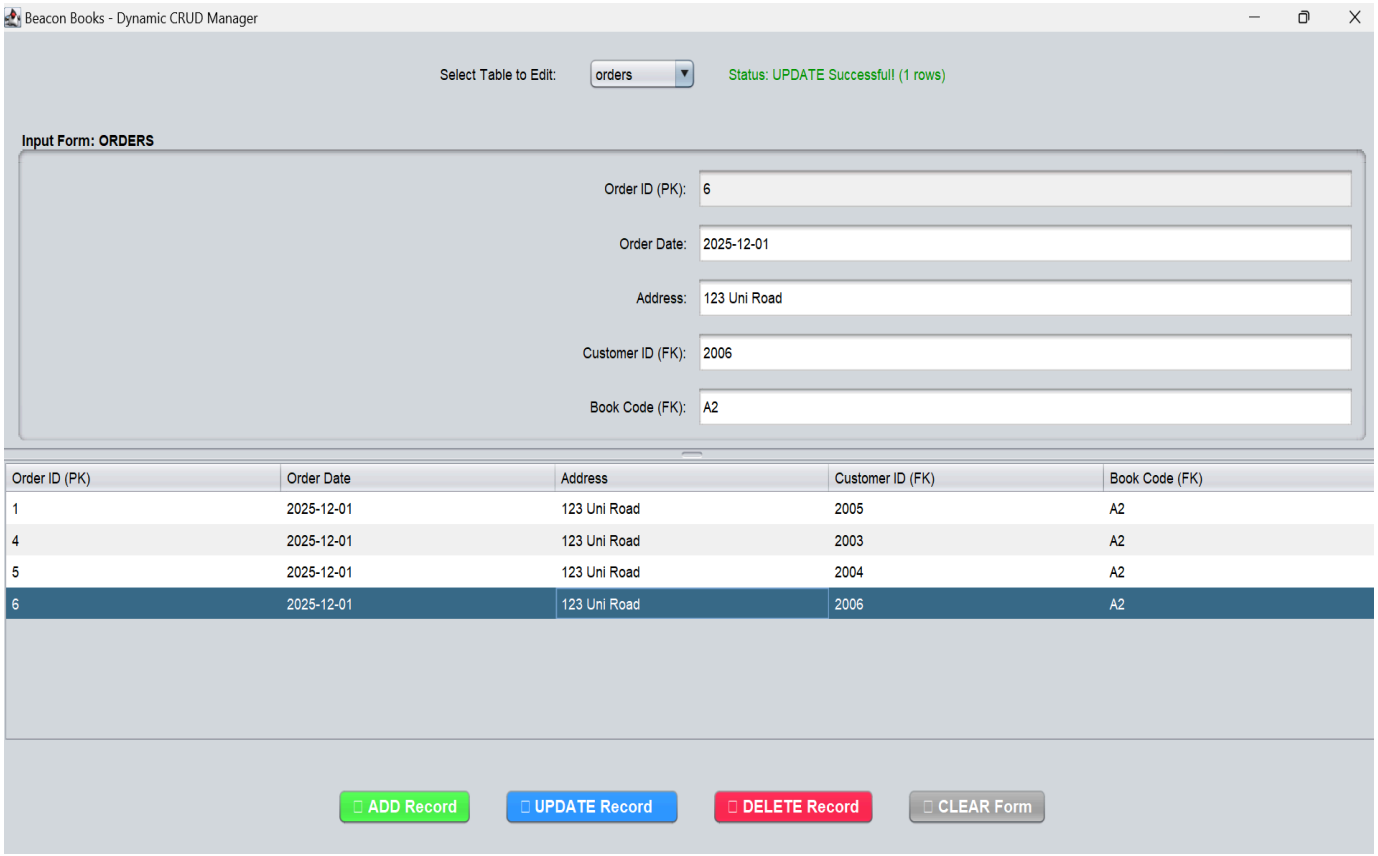
- Technology: Java Swing (JFrame, JPanel, JTable).
- Connection: The GUI uses the same JDBC parameters (DB_URL, USER, PASS) defined in the Entry class, demonstrating a consistent connection methodology across the entire application.
- Design: A BorderLayout was used to cleanly separate the three functional areas: the input fields (NORTH), the data display table (CENTER), and the action buttons (SOUTH).

Key CRUD Functionality

1. READ/Display:
    - The loadData() method is called immediately on startup and after every operation.
    - It executes SELECT * FROM orders and populates the data into a central JTable using a DefaultTableModel.
2. User Selection (MouseListener):
    - A MouseListener is attached to the JTable. When the user clicks a row, the corresponding record's data is instantly transferred to the input text fields (JTextFields). This feature makes Update and Delete operations intuitive.
3. CREATE, UPDATE, and DELETE:
    - Data Integrity: All button actions utilize the common executeSQL() helper method. This method uses PreparedStatement for all SQL execution. This is a critical security measure against SQL injection, ensuring robust handling of user input for the Add, Update, and Delete operations.
    - ADD: Executes INSERT INTO orders (order_date, delivery_address, customer_id, book_code) VALUES (?, ?, ?, ?).
    - UPDATE: Executes UPDATE orders SET ... WHERE order_id=?. Requires selecting a row first.
    - DELETE: Executes DELETE FROM orders WHERE order_id=?. Requires user confirmation via JOptionPane.

GUI display:


Beacon Books - Dynamic CRUD Manager
Select Table to Edit:
book
Status: Editing table BOOK
orders
book
customer
category
purchase
review_audit
Input Form: BOOK
Book
Price:
Category Code (FK):
Delivery Date:
Book Code (PK)
Title
Price
Category Code (FK)
Delivery Date
A2 ASP.NET 250.00 25 WSD
A3 PHP 250.00 25 WSD
A4 JavaFX 350.00 25 WSD
B2 Oracle 750.00 50 DDM
B3 SQLS 750.00 50 DDM
C2 Law 250.00 25 NSF
C3 Forensics 350.00 25 NSF
C4 Networks 250.00 25 NSF
ADD Record
UPDATE Record
DELETE Record
CLEAR Form


Beacon Books - Dynamic CRUD Manager
Select Table to Edit:
orders
Status: UPDATE Successful! (1 rows)
Input Form: ORDERS
Order ID (PK):
6
Order Date:
2025-12-01
Address:
123 Uni Road
Customer ID (FK):
2006
Book Code (FK):
A2
Order ID (PK)
Order Date
Address
Customer ID (FK)
Book Code (FK)
1 2025-12-01 123 Uni Road 2005 A2
4 2025-12-01 123 Uni Road 2003 A2
5 2025-12-01 123 Uni Road 2004 A2
6 2025-12-01 123 Uni Road 2006 A2
ADD Record
UPDATE Record
DELETE Record
CLEAR Form

5 Conclusion

The BeaconBooksDB system implementation is a successful response to all coursework requirements of the 5501SEPA module since it creates both a solid database backend and operational client interface. The database layer is based on the rigid principles of normalization and integrity with the use of composite keys and the with check option clause to guarantee high quality of data. The procedural requirements were also met by developing an automated schedulebookdeliveries procedure and security audit trigger. Moreover, inconsistencies in logic between the sample and the specification were eliminated due to the adherence to the coursework text to facilitate technical correctness. The Java console application, which goes together with the database, validates a successful JDBC connection, functional GUI and is capable of successfully communicating with the various parts of these databases, especially in its current driver settings and security measures. The combination of these factors proves a holistic, secure and technically correct approach to the management system of online bookstores.