# DBMS Review 2, Review 3

Functions:

- borrowed_count()
```
DECLARE cnt INT;
SELECT COUNT(*) INTO cnt
FROM BORROW
WHERE Student_ID = stu_id
AND Status = 'Issued';
RETURN cnt;
```

- is_book_available()
```
DECLARE s VARCHAR(20);
 SELECT Status INTO s
  FROM BOOK
    WHERE Book_ID = bookid
    LIMIT 1;
    IF s = 'Available' THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
```

- overdueby()
```
RETURN DATEDIFF(CURDATE(), due_date);
```

- total_books_in_genre()
```
RETURN (
    SELECT COUNT(*)
    FROM book
    WHERE genre = genre_name
)
```

- active_staff_count()
```
   RETURN (
    SELECT COUNT(*)
    FROM staff
    WHERE status = 'active'
)
```

Stored Procedures:

- get_books_borrowed_by_student()

```
BEGIN
   SELECT
      B.Book_ID,
      B.Title,
      B.Author,
      B.Publisher,
      B.Year_Published,
      B.Genre,
      Br.Issue_Date,
      Br.Due_Date,
      Br.Status AS Borrow_Status
   FROM BOOK B
   JOIN BORROW Br ON B.Book_ID = Br.Book_ID
   WHERE Br.Student_ID = studentId
     AND TRIM(Br.Status) IN ('not returned', 'issued', 'borrowed');
END
```

- get_books_borrowed_with_overdue()

```
BEGIN
   SELECT
      B.Title AS Book_Title,
      overdueby(Br.Due_Date) AS Overdue //Uses the function in the procedure
   FROM BOOK B
   JOIN BORROW Br ON B.Book_ID = Br.Book_ID
   WHERE Br.Student_ID = studentId
     AND TRIM(Br.Status) IN ('not returned', 'issued', 'borrowed');
END
```

- add_new_book()

```
 IN p_title VARCHAR(100),
 IN p_author VARCHAR(100),
IN p_genre VARCHAR(50),
IN p_publish_date DATE,
IN p_cost DECIMAL(8,2)
 )
   BEGIN
  INSERT INTO book (title, author, genre, publish_date, cost, status)
  VALUES (p_title, p_author, p_genre, p_publish_date, p_cost, 'available');
```

```
-    get_active_staff_list()
            BEGIN
               SELECT staff_id, first_name, last_name, position
               FROM staff
               WHERE status = 'active';
            END

-   get_currently_borrowed_books()
             BEGIN
               SELECT b.Book_ID, b.Title, b.Author, b.Genre,
                    br.Student_ID, br.Staff_ID, br.Issue_Date, br.Due_Date, br.Status
               FROM book b
               JOIN borrow br ON b.Book_ID = br.Book_ID
               WHERE br.Status = 'Issued';
            END
```

Triggers:

```
| TRIGGER_NAME       | Event  | Table_Name | Timing | Definition

-   | after_borrow_insert | INSERT | BORROW     | AFTER  | BEGIN
       UPDATE book
       SET Status = 'Issued'
       WHERE Book_ID = NEW.Book_ID;
        END

-   | after_borrow_return | UPDATE | BORROW     | AFTER  | BEGIN
       IF NEW.Status != 'Issued' THEN
       UPDATE book
       SET Status = 'Available'
       WHERE Book_ID = NEW.Book_ID;
       END IF;
       END
-   | after_book_insert   | INSERT | BOOK       | AFTER  | BEGIN
       INSERT INTO genre_count (Genre, Count)
       VALUES (NEW.Genre, 1)
       ON DUPLICATE KEY UPDATE Count = Count + 1;
    END

-   | before_book_delete  | DELETE | BOOK       | BEFORE | BEGIN
       IF OLD.Status = 'Issued' OR OLD.Status = 'Borrowed' THEN
```

```
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot delete a book that is currently issued or
borrowed';
    END IF;
END
```

-
  ```
  | after_staff_insert  | INSERT | staff      | AFTER  | BEGIN
      -- Ensure status is set to Active if not provided
      IF NEW.Status IS NULL OR NEW.Status = '' THEN
          UPDATE staff
          SET Status = 'Active'
          WHERE Staff_ID = NEW.Staff_ID;
      END IF;
  END
  ```

-
  ```
  | before_borrow_limit | INSERT | BORROW     | BEFORE | BEGIN
      DECLARE borrow_count INT;
      -- Count how many books this student currently has issued
      SELECT COUNT(*)
      INTO borrow_count
      FROM borrow
      WHERE Student_ID = NEW.Student_ID
        AND Status = 'Issued';
      -- Prevent borrowing if already 3 or more books
      IF borrow_count >= 3 THEN
          SIGNAL SQLSTATE '45000'
          SET MESSAGE_TEXT = 'Cannot borrow more than 3 books at a time';
      END IF;
  END |
  ```