Stage 5: Final Demo

Group: Ace_DB

1. **Stored Procedures and a Trigger**

    a. **Stored procedure 1: Title for Upvotes**

    This SP will give users different titles in terms of the total number of upvotes they gave others' reviews:

    - 0 Upvotes: Starter
    - 1 - 3 Upvotes: Intermediate
    - 3 - 5 Upvotes: Advanced
    - All other cases (e.g. NULL): Beginner

    This SP will make it easier for users of the web application to get a sense of activity levels, and hierarchies used to classify those levels (an analogy to the Daily Active Users measurement). Specifically, this will allow users to be informed about how many upvotes they need "give" in order to achieve a certain title or level.

```
DELIMITER //
CREATE PROCEDURE User_upvote_5more_16()
BEGIN

    DECLARE userName VARCHAR(255);
    DECLARE userID INT;
    DECLARE Title VARCHAR(255);
    DECLARE theNumUpvote VARCHAR(50);
    DECLARE theAverageScore REAL;


    DECLARE exit_loop BOOLEAN DEFAULT FALSE;

    DECLARE cur CURSOR FOR

    -- (
    -- SELECT U.UserID, U.Username, SUM(new1.numUv) AS thisNumUpvote
    -- FROM (SELECT PostID, COUNT(UserID) AS numUv
    --         FROM Upvoted
    --         GROUP BY PostID ) AS new1
    --         JOIN Review R ON new1.PostID= R.PostID JOIN User U ON U.UserID = R.UserID
    -- GROUP BY U.UserID
    -- );

    -- (
    -- SELECT U1.UserID, U1.Username, 0
    -- FROM User U1
    -- WHERE U1.UserID NOT IN (SELECT Uv1.UserID
    --                         FROM Upvoted Uv1)
    -- GROUP BY U1.UserID
    -- )
    -- UNION

    (
    SELECT U.UserID, U.Username, COUNT(Uv.PostID) AS thisNumUpvote
    FROM Upvoted Uv RIGHT OUTER JOIN User U ON Uv.UserID = U.UserID
    GROUP BY U.UserID
    );

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop = TRUE;
    /*create a new table part*/
    DROP TABLE IF EXISTS FinalTable;

    CREATE TABLE FinalTable (
        UserID_ INT Primary Key,
        userName_ VARCHAR(255),
        theNumUpvote_ REAL,
        Title_ VARCHAR(50)
    );

```

```
    OPEN cur;
    loop1: LOOP
        FETCH cur INTO userID, userName, theNumUpvote;
        IF userID = NULL THEN
            LEAVE loop1;
        END IF;
        IF exit_loop THEN
            LEAVE loop1;
        END IF; /*do not miss this END IF*/

        IF theNumUpvote = 0 THEN
            SET Title = 'starter';
          ELSEIF theNumUpvote >=1 AND theNumUpvote < 3 THEN
            SET Title = 'intermediate';
          ELSEIF theNumUpvote >=3 AND theNumUpvote < 5 THEN
            SET Title = 'advanced';
          ELSE
            SET Title = 'beginner';
        END IF; /*split the word!*/

        INSERT IGNORE INTO FinalTable VALUES (userID, userName, theNumUpvote, Title);  /*ignore
            duplicates*/
      END LOOP loop1;
    CLOSE cur;
    /*return the NetId and Course_Load_Status of the NewTable from within the Stored Procedure*/
    SELECT Title_, avg(theNumUpvote_) AS avg_uv
    FROM FinalTable
    GROUP BY Title_
    Having Title_ = "intermediate"
    UNION
    SELECT Title_, avg(theNumUpvote_) AS avg_uv
    FROM FinalTable
    GROUP BY Title_
    Having Title_ = "advanced";

    -- SELECT Title_, count(*) AS avg_uv
    -- FROM FinalTable
    -- GROUP BY Title_

    -- SELECT Title_, avg(theNumUpvote_)
    -- FROM FinalTable
    -- GROUP BY Title_
    -- LIMIT 15;
END;

```

```
mysql> CALL User_upvote_5more_16();
    -> //
+--------------+--------------------+
| Title_       | avg_uv             |
+--------------+--------------------+
| intermediate | 1.1879061881887538 |
| advanced     | 3.3732718894009217 |
+--------------+--------------------+
2 rows in set (21.51 sec)

Query OK, 0 rows affected (21.51 sec)

mysql>
```

b. Stored Procedure 2: Title for Reviews

This SP is similar to SP 1, but differentiated in that it will assign a level or title to users based on the number of reviews they have given:

- 0 Reviews: Bronze
- 1-5 Reviews: Silver
- 5+ Reviews: Gold
- All other cases: Unknown

This will easily communicate to the web application's users, at a glimpse, activity levels and hierarchies used to classify those levels (an analogy to the Daily Active Users measurement). Specifically, this will allow users to be informed about how many reviews they need to write in order to achieve a certain title or level.

```
DELIMITER //
CREATE PROCEDURE User_review_16()
BEGIN

    DECLARE userName VARCHAR(255);
    DECLARE userID INT;
    DECLARE Title VARCHAR(255);
    DECLARE theNumRev VARCHAR(50);
    DECLARE theAverageScore REAL;


    DECLARE exit_loop BOOLEAN DEFAULT FALSE;

    DECLARE cur CURSOR FOR
    (
    SELECT U.UserID, U.Username, COUNT(R.PostID) AS thisNumRev
    FROM Review R JOIN User U ON R.UserID = U.UserID
    GROUP BY U.UserID
    );

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop = TRUE;
    /*create a new table part*/
    DROP TABLE IF EXISTS FinalTable;

    CREATE TABLE FinalTable (
        UserID_ INT Primary Key,
        userName_ VARCHAR(255),
        theNumRev_ REAL,
        Title_ VARCHAR(50)
    );

    OPEN cur;
    loop1: LOOP
        FETCH cur INTO userID, userName, theNumRev;
        IF userID = NULL THEN
            LEAVE loop1;
        END IF;
        IF exit_loop THEN
            LEAVE loop1;
        END IF; /*do not miss this END IF*/

        IF theNumRev = 0 THEN
            SET Title = 'bronze';
          ELSEIF theNumRev >=1 AND theNumRev <5 THEN
            SET Title = 'silver';
          ELSEIF theNumRev >= 5 THEN
            SET Title = 'gold';
          ELSE
            SET Title = 'unknown';
        END IF; /*split the word!*/
```

```
51
52          INSERT IGNORE INTO FinalTable VALUES (userID, userName, theNumRev, Title);  /*ignore
                duplicates*/
53        END LOOP loop1;
54      CLOSE cur;
55      /*return the NetId and Course_Load_Status of the NewTable from within the Stored Procedure*/
56      -- SELECT userName_,  Title_, UserID_, theNumRev_
57      -- FROM FinalTable
58      -- ORDER BY Title_, userName_
59      -- Limit 10;
60
61      SELECT Title_, avg(theNumRev_) AS avg_uv
62      FROM FinalTable
63      GROUP BY Title_
64      Having Title_ = "silver"
65      UNION
66      SELECT Title_, avg(theNumRev_) AS avg_uv
67      FROM FinalTable
68      GROUP BY Title_
69      Having Title_ = "gold";
70
71  END;
72
```

```
mysql> CALL User_review_16()//
+--------+--------------------+
| Title_ | avg_uv             |
+--------+--------------------+
| silver | 1.4244640605296344 |
| gold   |  8.049418604651162 |
+--------+--------------------+
2 rows in set (21.19 sec)

Query OK, 0 rows affected (21.19 sec)
```

c. **Trigger**: The trigger implemented will prevent users from updating user information with erroneous values. If an overly high updated age is entered (in our current trigger, we are using any number above 150 years of age), the user's updated information will immediately go into a different table called "User_info" before the update takes effect in the authoritative "User" table. Alongside this, their region will be updated to "Heaven", indicating that no living person has a physical presence at such a high age.

```
DELIMITER //
CREATE TRIGGER update_check
   BEFORE UPDATE ON User
```

```
    FOR EACH ROW
BEGIN
    IF new.Age > 150  THEN
        INSERT INTO User_info values (new.UserID, new.Age, new.Region);
        SET new.Region = "Heaven";
    END IF;
END//
```

2. **How did the creative component add extra value to your application?**
   At this time, we have not created a creative component.

3. **How would you want to further improve your application? In terms of database design and system optimization?**
   Currently, there is no prevention of dirty read, unrepeatable read or write-write conflicts. In the future, an important prospective next step would include implementing a design to this system which prevents these types of concurrent executions.

4. **What were the challenges you faced when implementing and designing the application? How was it the same/different from the original design?**
   Front-end design and visualization could have been further polished with a greater amount of time, and more features could have been included with it. In previous stages, we had planned for a better user-interface for this system, implementing functionalities such as a comparison dashboard, which would have been helpful for comparing key metrics between two different games in a visually accessible and pleasing presentation.This part of our plan, unfortunately, was not realized but would be a beneficial addition.

5. **If you were to include a NoSQL database, how would you incorporate it into your application?**
   We believe that incorporating Neo4j would be possible since our data is highly connected. We have five relational tables, which is a bit tedious when running advanced queries. In Neo4j, the need to JOIN every table could be eliminated. More specifically and in example, in Neo4j, it would be helpful for us to store the User table and all related data to simplify and streamline retrieval from such advanced queries, since they are highly connected.