Dream to Stream
Stage 5 Final Demo

Stored Procedure + Trigger

trigger.sql

```
/*
   boundMovieRating and boundShowRating
   Triggers that allow for Update action using condition (IF Statement)
   on MovieRating and ShowRating.

   Sets bounds on:
        - if users set a value greater than 10 to set it to 10
        - if users set a value less than 0 to set it to 0
   This is done before any UPDATES.
*/

-- Delimiter used for GCP MySQL
DELIMITER //

CREATE TRIGGER boundMovieRating BEFORE UPDATE ON MovieRating FOR EACH ROW
BEGIN
   IF NEW.value > 10 THEN
        SET NEW.value = 10;
   ELSEIF NEW.value < 0 THEN
        SET NEW.value = 0;
   END IF;
END;
```

```
CREATE TRIGGER boundShowRating BEFORE UPDATE ON ShowRating FOR EACH ROW BEGIN
    IF NEW.value > 10 THEN
        SET NEW.value = 10;
    ELSEIF NEW.value < 0 THEN
        SET NEW.value = 0;
    END IF;
END;


-- Delimiter used for GCP MySQL
DELIMITER ;
```

stored_procedure.sql

```
/*
 get_rating Procedure that gets ratings given two userIds (INT) as parameters

 This will then run through all Movies and Shows both users have watched
 and compare those that both have placed reviews for and store them as cursors
(movie_cursor and show_cursor).

 Next, it will loop through each and concat out Movies/Shows that both have watched
where:
   - userId1 rated higher than userId2
   - userId2 rated higher than userId1
   - Both users rated them the same

 This would allow for users to have a better understanding of "Mesh" where they can
see how compatible they are
 in terms of their Show/Movie likings.
*/

-- Delimiter used for GCP MySQL
DELIMITER //

CREATE PROCEDURE get_ratings(IN userId1 INT, IN userId2 INT)
BEGIN
 DECLARE movie_name VARCHAR(255);
 DECLARE movie_rating1 INT;
 DECLARE movie_rating2 INT;
```

```
 DECLARE show_name VARCHAR(255);
 DECLARE show_rating1 INT;
 DECLARE show_rating2 INT;
 DECLARE exit_loop BOOLEAN DEFAULT FALSE;
  -- USING JOIN (ADVANCED QUERY 1) to get Movies that both users have reviewed from MovieRating
 DECLARE movie_cursor CURSOR FOR
   SELECT MR.name, MR.value as rating1, MR2.value as rating2
   FROM MovieRating MR
   INNER JOIN MovieRating MR2 ON MR.name = MR2.name
   WHERE MR.id = userId1 AND MR2.id = userId2;

 -- USING SUBQUERY (ADVANCED QUERY 2) to get Shows that both users have reviewed from ShowRating
 DECLARE show_cursor CURSOR FOR
   SELECT SR.name,
     (
       SELECT value FROM ShowRating
       WHERE id = userId1 AND name = SR.name
     ) as rating1,
     (
       SELECT value FROM ShowRating
       WHERE id = userId2 AND name = SR.name
     ) as rating2
   FROM ShowRating SR
   WHERE SR.id IN (userId1, userId2)
   GROUP BY SR.name -- Group by ShowRating.Name (ADVANCED QUERY)
```

```
    HAVING COUNT(DISTINCT id) = 2;


 DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop = TRUE;

  SET movie_rating1 = 0;

 SET movie_rating2 = 0;

 SET show_rating1 = 0;

 SET show_rating2 = 0;

  -- LOOP THROUGH MOVIE RATINGS --

 OPEN movie_cursor; -- USING CURSOR

 movie_loop: LOOP -- USING LOOPING STRUCTURE

   FETCH movie_cursor INTO movie_name, movie_rating1, movie_rating2;


   IF exit_loop THEN

     LEAVE movie_loop;

   END IF;


   -- Processing Movie Ratings USING CONTROL STRUCTURE (IF-Statement)

   IF movie_rating1 > movie_rating2 THEN

     SELECT CONCAT('User ', userId1, ' rated the Movie "', movie_name, '" higher than
User ', userId2, ' (', movie_rating1, ' vs ', movie_rating2, ')') AS comparison;

   ELSEIF movie_rating1 < movie_rating2 THEN

     SELECT CONCAT('User ', userId2, ' rated the Movie "', movie_name, '" higher than
User ', userId1, ' (', movie_rating2, ' vs ', movie_rating1, ')') AS comparison;

   ELSE

     SELECT CONCAT('Both users rated "', movie_name, '" equally (', movie_rating1,
')') AS comparison;

   END IF;
```

```
    END LOOP;

     CLOSE movie_cursor;

      -- LOOP THROUGH SHOW RATINGS --

    SET exit_loop = FALSE;

    OPEN show_cursor; -- USING CURSOR

    show_loop: LOOP -- USING LOOPING STRUCTURE

      FETCH show_cursor INTO show_name, show_rating1, show_rating2;


      IF exit_loop THEN

        LEAVE show_loop;

      END IF;


      -- Processing Show Ratings USING CONTROL STRUCTURE (IF-Statement)


      IF show_rating1 > show_rating2 THEN

        SELECT CONCAT('User ', userId1, ' rated the Show "', show_name, '" higher than
User ', userId2, ' (', show_rating1, ' vs ', show_rating2, ')') AS comparison;

      ELSEIF show_rating1 < show_rating2 THEN

        SELECT CONCAT('User ', userId2, ' rated the Show "', show_name, '" higher than
User ', userId1, ' (', show_rating2, ' vs ', show_rating1, ')') AS comparison;

      ELSE

        SELECT CONCAT('Both users rated "', show_name, '" equally (', show_rating1, ')')
AS comparison;

      END IF;


    END LOOP;

     CLOSE show_cursor;

    END //
```

```
-- Delimiter used for GCP MySQL

DELIMITER ;
```

Presentation Breakdown

Divya & Mohammad:

- Using GCP to host the DB and VMs
- Data sources used: Netflix, Hulu, Disney+, Prime Video
- Demo of the CRUD operations within the application
- Advanced Queries

Anthony

- Explain your choice for the advanced database program and how it is suitable for your application. For example, if you chose a stored procedure+trigger, explain how this choice is suitable for your application.
- How did the creative element add extra value to your application?

Rohan:

- Trigger usage
- If you were to include a NoSQL database, how would you incorporate it into your application?
    - MongoDB in case people want to do longer writeups (Paragraphs instead of having max characters)
    - Handling logger data types like casts, directors and Movie description

Team Reflections:

Anthony: What were the challenges you faced when implementing and designing the application?

Rohan: How was it the same/different from the original design? Integration NoSQL

Divya: How would you want to further improve your application?

Mohammad: In terms of database design and system optimization?

Reflection Questions

- What were the challenges you faced when implementing and designing the application?

    - General knowledge of using MySQL syntax and uploading data up to GCP
        - Triggers, Stored Procedures, and delimiters (understanding it better) - we had a lock that occurred that prevented us from updating the tables.
        - Loading the data (breaking up by platforms)
        - Making design decisions without understanding how it could impact us down the line
    - Understanding Node.js and utilizing it within GCP without being able to directly connect it to Github due to access limitations.
    - Front end development and how to integrate SQL directly into the tables and linking pages together.
- How was it the same/different from the original design?
    - Initially, the goal was to decide to keep everything in its own Platform table, but the team realized that it would be easier to merge Movies and Shows into its own tables and have control of the column names as it would allow for the team to have better visibility of the data all in a centralized table for Movies and Shows.
        - After, the design has been the same as we have used all of the tables mentioned within the application to add users, let them view Movies/Shows from the platforms combined, Updating and Deleting ratings, and signing in & keeping their reviews under their account.
- How would you want to further improve your application?
    - Utilization of the location data provided to show heat maps of what is available. Would be of interest to users.
    - For Stored Procedure Mesh, showing both user's names instead of IDs
- In terms of database design and system optimization?
    - Combining some of the tables we had together (e.g. MovieRatings and ShowRatings) since the columns and fields are the same. Instead, the 'Ratings' table would have an extra column identifying if the rating is for Movies or Shows. The result would reduce the repeated code and logic needed from the application (no extra if statements to loop through).
    - This would also add capabilities to allow for us to add queries that would just run on one table instead of two for the recommendation because the application could show all results from that one table and utilize the Show/Movie identification column