**DELIVERABLE 4: FINAL PROJECT**

**DSBA-HCIP 6201 Databases for Data Scientists**

**TEAM MEMBERS:**

Julian Martin, Mo Diallo, Sophia Gucciardi, Jyothsna Ravipalli, Madhuri Yerram.

**Create 3 advanced views that use a mix of the following:**

**nested query, sorting, and conditional statements**

**group by and summary functions**

**WINDOW functions and CTEs.**

**VIEW 1:**

A screenshot of a computer

Description automatically generated

**VIEW 2:**

**A screenshot of a computer

Description automatically generated**

**VIEW 3:A screenshot of a computer

Description automatically generated**

**View 1:**The code creates a view `average\_user\_ratings\_comparison` comparing average user ratings among seasons. It satisfies specified criteria through CTEs, nested queries, conditional statements, and group by with summary functions. CTE’s `season\_avg\_ratings` and `season\_comparison` calculate average ratings and compare them. Nested queries in CTEs aggregate data from `user\_ratings` table. Conditional CASE statement in `season\_comparison` labels comparisons as 'Higher', 'Lower', or 'Equal'. Results are implicitly sorted by season numbers. `season\_avg\_ratings` uses GROUP BY and AVG () to calculate average ratings per season. The code does not utilize window functions. It effectively analyzes user engagement with TV series across seasons, providing insights on rating variations over time.

**View 2:**

The code creates a view `top\_episodes\_user\_ratings` calculating average user ratings for each episode and ranking them. It satisfies criteria by utilizing CTE for calculating ratings and serving as an intermediary result set. CTE includes a nested query joining tables to calculate average ratings. Data in CTE is grouped by season and episode number, and AVG () function calculates average user ratings. Main query selects data from CTE, groups result by season and episode number, calculates overall average user ratings, and orders results by season and rating. Overall, the view uses CTEs, nested queries, group by, summary functions, and sorting to rank episodes based on user ratings across seasons, aiding stakeholders in analysis and decision-making for TV series.

**View 3:**

The code creates a view `comment\_activity\_by\_episode` meeting specified criteria by utilizing CTEs, nested queries, conditional statements, group by with summary functions, and window functions. CTE `CommentCounts` calculates comments per episode. A CASE statement in the main SELECT query acts as a conditional statement. Grouping is done by episode\_num\_overall in the CTE and episode labeling based on comment count comparison with the average. The code calculates the average comment count across all episodes using the AVG () window function without partitioning, providing insights on comment activity per episode.