Please read `group_12_trigger_test_cases.pdf`

DETAILS TRIGGERS

Final Project - FIFA World Cup

GROUP 12 - ETI2V.IB

Version 1.2

Please read `group_12_trigger_test_cases.pdf`

Please read `group_12_trigger_test_cases.pdf`

GUIDELINE TO USE THE DATABASE

There are two ways to using the database:

1. If you have the `messy_CVS` database in your SQL Server, you can use the `group_12_CVS_script_NEW.sql` to create and insert data into `group12_final` database. The logic of the `.sql` file is:
    1. Create database `group12_final IF NOT EXISTS`
    2. Create the schema `WC`
    3. Create the tables
    4. Create triggers
    5. Finally, insert data from `messy_CVS` database.
2. The second way is using `group_12_DBScript_NEW.sql` which:
    1. Create database `group12_final`
    2. Create the schema `WC`
    3. Insert data into all tables
    4. Finally, create triggers You can chose 1 way to insert and use the test cases below.

> `group_12_triggers.sql` file only listed all the triggers group 12 have. User only need to run `group_12_CVS_script_NEW.sql` or `group_12_DBScript_NEW.sql` to use `group12_final` database.

TRIGGER TEST CASE ISSUES DURING THE DEMO

The reason why Group 12's demo failed in testing the `group_standing` trigger is not because of the triggers themselves, but because we missed the syntax:

```
USE group12_final;
GO
```

Which uses `master` database to insert tables and triggers into the `master` database.

GUIDELINE TO USE TRIGGER TEST CASE

This test case can be use step by step:

1. Use queries in
    1. Group Standing test cases
    2. Tournament Statistics test cases
    3. Award Winners test cases to check the value for each test case.
2. Using data Testing data section to insert new records for testing
3. Check all queries from the test cases again to see difference.
4. Using Delete Test records section to delete test data.

Please read `group_12_trigger_test_cases.pdf`

TEST CASES:

You can find the test case in file `group_12_trigger_test_case.pdf`, which explains the way to insert data into the database and the test case to testing the triggers.

EXPLAIN THE TRIGGERS TO GENERATE TOURNAMENT STATISTICS

Created triggers to insert data into **player_statistics**, **team_statistics**, and **tournament_standing**, then data into the **tournament_statistics** table.

Trigger insert data into **team_statistics** and **player_statistics:**

Trg_after_group_standing_insert_team_statistics

- After inserting data into the **group_standing** table, it means knowing which team belong to which group during that tournament.
- Insert new records for each team (check if they exist or not in **team_statistics** table)

```
CREATE TRIGGER [WC].[trg_after_group_standing_insert_team_statistics]
ON  [WC].[group_standing]
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
       -- Insert into team_statistic table
       INSERT INTO  WC.team_statistics(team_id, tournament_id, tournaments)
    SELECT i.team_id, i.tournament_id,
           (SELECT COUNT(DISTINCT s.tournament_id)
            FROM  WC.group_standing s
            WHERE s.team_id = i.team_id)
    FROM inserted i
    WHERE NOT EXISTS (
        SELECT 1
        FROM  WC.team_statistics ts
        WHERE ts.team_id = i.team_id
        AND ts.tournament_id = i.tournament_id
    );
       -- Update the tournaments in team_statistic table for the inserted squad
       UPDATE ts
    SET ts.tournaments =
        (SELECT COUNT(DISTINCT s.tournament_id)
         FROM  WC.group_standing s
         WHERE s.team_id = ts.team_id)
    FROM WC.team_statistics ts
    JOIN inserted i
    ON ts.team_id = i.team_id;
END;
GO
```

Please read `group_12_trigger_test_cases.pdf`

Trg_after_squad_insert_player_statistics

- After inserting data into the **squad** table (know which teams and players belong to which tournament).
- Inserting data into **player_statistics** table if not exists.

```
CREATE TRIGGER [WC].[trg_after_group_standing_insert_team_statistics]
ON  [WC].[group_standing]
AFTER INSERT
AS
BEGIN
       -- Insert into team_statistic table
       INSERT INTO  WC.team_statistics(team_id, tournament_id, tournaments)
    SELECT i.team_id, i.tournament_id,
           (SELECT COUNT(DISTINCT s.tournament_id)
            FROM  WC.group_standing s
            WHERE s.team_id = i.team_id)
    FROM inserted i
    WHERE NOT EXISTS (
        SELECT 1
        FROM  WC.team_statistics ts
        WHERE ts.team_id = i.team_id
        AND ts.tournament_id = i.tournament_id
    );
       -- Update the tournaments in team_statistic table for the inserted squad
       UPDATE ts
    SET ts.tournaments =
        (SELECT COUNT(DISTINCT s.tournament_id)
         FROM  WC.group_standing s
         WHERE s.team_id = ts.team_id)
    FROM WC.team_statistics ts
    JOIN inserted i
    ON ts.team_id = i.team_id;
END;
GO
```

Trg_match_update_team_statistics

This trigger insert and update attribute **matches** in **team_statistics**

```
CREATE TRIGGER [WC].[trg_match_update_team_statistics]
ON  [WC].[match]
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
       UPDATE stats
    SET stats.matches = combined.matches
    FROM (
        -- Group and count the matches played by each team in each tournament
```

Please read `group_12_trigger_test_cases.pdf`

```
        SELECT A.tournament_id, A.team_id, COUNT(DISTINCT A.match_id) AS matches
                    FROM  (
                           SELECT tournament_id, home_team_id AS team_id, match_id
                           FROM  WC.match
                           UNION ALL
                           SELECT tournament_id, away_team_id AS team_id, match_id
                           FROM  WC.match
                    ) AS A
              GROUP BY A.tournament_id, A.team_id
    ) AS combined
    INNER JOIN  WC.team_statistics AS stats
        ON stats.tournament_id = combined.tournament_id
        AND stats.team_id = combined.team_id;

       -- INSERT statement to add new rows if they don't exist
    INSERT INTO  WC.team_statistics (tournament_id, team_id, matches)
    SELECT combined.tournament_id, combined.team_id, combined.matches
    FROM (
        SELECT A.tournament_id, A.team_id, COUNT(DISTINCT A.match_id) AS matches
                    FROM  (
                           SELECT tournament_id, home_team_id AS team_id, match_id
                           FROM  WC.match
                           UNION ALL
                           SELECT tournament_id, away_team_id AS team_id, match_id
                           FROM  WC.match
                    ) AS A
              GROUP BY A.tournament_id, A.team_id
    ) AS combined
    WHERE NOT EXISTS (
        SELECT 1
        FROM  WC.team_statistics stats
        WHERE stats.tournament_id = combined.tournament_id
        AND stats.team_id = combined.team_id
    );

END;
GO
```

Trg_update_player_statistics

When player appeared in a match, then we update attribute **matches** in player_statistics table.

```
CREATE TRIGGER trg_update_player_statistics
ON WC.player_appearance
AFTER INSERT, UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    -- Update existing records in the player_statistics table
    UPDATE stats
    SET stats.matches = combined.matches
```

Please read `group_12_trigger_test_cases.pdf`

```
    FROM WC.player_statistics AS stats
    INNER JOIN (
              SELECT
            TEMP.tournament_id,
            TEMP.player_id,
            COUNT(DISTINCT TEMP.match_id) AS matches
        FROM (
            SELECT
                PA.player_id,
                'WC-' + SUBSTRING(PA.match_id, 3, 4) AS tournament_id,
                PA.match_id
            FROM WC.player_appearance AS PA
            INNER JOIN WC.squad AS S ON PA.player_id = S.player_id
            AND 'WC-' + SUBSTRING(PA.match_id, 3, 4) = S.tournament_id
        ) AS TEMP
        GROUP BY
            TEMP.tournament_id,
            TEMP.player_id
       ) AS combined
        ON stats.tournament_id = combined.tournament_id
        AND stats.player_id = combined.player_id;

    -- Insert new records if they don't exist
    INSERT INTO WC.player_statistics (tournament_id, player_id, matches)
    SELECT combined.tournament_id, combined.player_id, combined.matches
    FROM (
              SELECT
            TEMP.tournament_id,
            TEMP.player_id,
            COUNT(DISTINCT TEMP.match_id) AS matches
        FROM (
            SELECT
                PA.player_id,
                'WC-' + SUBSTRING(PA.match_id, 3, 4) AS tournament_id,
                PA.match_id
            FROM WC.player_appearance AS PA
            INNER JOIN WC.squad AS S ON PA.player_id = S.player_id
            AND 'WC-' + SUBSTRING(PA.match_id, 3, 4) = S.tournament_id
        ) AS TEMP
        GROUP BY
            TEMP.tournament_id,
            TEMP.player_id
       ) AS combined
    WHERE NOT EXISTS (
        SELECT 1
        FROM WC.player_statistics AS stats
        WHERE stats.tournament_id = combined.tournament_id
        AND stats.player_id = combined.player_id
    );
END;
GO
```

Please read `group_12_trigger_test_cases.pdf`

Trg_update_statistic_table_booking_attributes

This trigger update attributes relate to booking event (yellow_cards, red_cards, sent_off) for 3 statistic table

```
CREATE TRIGGER [WC].[trg_update_statistic_table_booking_attributes]
ON  [WC].[booking]
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    -- Use try-catch for error handling
    BEGIN TRY

        -- Declare a temporary table to hold card statistics
        CREATE TABLE #TempTeamCard (
            tournament_id NVARCHAR(100),
            team_id NVARCHAR(100),
            yellow_cards INT,
            red_cards INT
        );

        -- Insert aggregated yellow and red card data into the temporary table
        INSERT INTO #TempTeamCard  (tournament_id, team_id, yellow_cards, red_cards)
        SELECT
            TEMP.tournament_id,
            TEMP.team_id,
            SUM(TEMP.y) AS yellow_cards,
            SUM(TEMP.r) AS red_cards
        FROM
        (
            SELECT
                M.tournament_id,
                M.match_id,
                ME.team_id,
                SUM(CASE WHEN B.yellow_card = 1 THEN 1 ELSE 0 END) AS y,
                SUM(CASE WHEN B.red_card = 1 THEN 1 ELSE 0 END) AS r
            FROM
                 WC.[match] AS M
            INNER JOIN
                 WC.match_event AS ME
            ON M.match_id = ME.match_id
            INNER JOIN
                 WC.booking AS B
            ON ME.event_id = B.event_id
            GROUP BY M.tournament_id, M.match_id, ME.team_id
        ) AS TEMP
        GROUP BY TEMP.tournament_id, TEMP.team_id;

        -- Update the team_statistic table with yellow and red card data
        UPDATE GS
        SET
            GS.yellow_cards = TEMP.yellow_cards,
            GS.red_cards = TEMP.red_cards
        FROM
             WC.team_statistics AS GS
        INNER JOIN
            #TempTeamCard AS TEMP ON
```

Please read `group_12_trigger_test_cases.pdf`

```
            GS.tournament_id = TEMP.tournament_id
            AND GS.team_id = TEMP.team_id;

        -- Drop the temporary table after use
        DROP TABLE #TempTeamCard;

        -- Declare a temporary table to hold card statistics
        CREATE TABLE #TempPlayerCard (
            tournament_id NVARCHAR(100),
            player_id NVARCHAR(100),
            yellow_cards INT,
            red_cards INT,
                      sent_offs INT
        );

        -- Insert aggregated yellow and red card data into the temporary table
        INSERT INTO #TempPlayerCard (tournament_id, player_id, yellow_cards, red_cards,
sent_offs)
        SELECT
            TEMP.tournament_id
            , TEMP.player_id
            , SUM(TEMP.y) AS yellow_cards
            , SUM(TEMP.r) AS red_cards
            , SUM(TEMP.s) AS sent_offs
        FROM
        (
            SELECT
                M.tournament_id
                , M.match_id
                , ME.player_id
                , SUM(CASE WHEN B.yellow_card = 1 THEN 1 ELSE 0 END) AS y
                , SUM(CASE WHEN B.red_card = 1 THEN 1 ELSE 0 END) AS r
                            , SUM(CASE WHEN B.sent_off = 1 THEN 1 ELSE 0 END) AS s
            FROM
                 WC.[match] AS M
            INNER JOIN
                 WC.match_event AS ME
            ON M.match_id = ME.match_id
            INNER JOIN
                 WC.booking AS B
            ON ME.event_id = B.event_id
            GROUP BY M.tournament_id, M.match_id, ME.player_id
        ) AS TEMP
        GROUP BY TEMP.tournament_id, TEMP.player_id;

        -- Update the team_statistic table with yellow and red card data
        UPDATE GS
        SET
            GS.yellow_cards = TEMP.yellow_cards
            , GS.red_cards = TEMP.red_cards
            , GS.sent_offs = TEMP.sent_offs
        FROM
             WC.player_statistics AS GS
        INNER JOIN
            #TempPlayerCard AS TEMP
              ON
```

Please read `group_12_trigger_test_cases.pdf`

```
            GS.tournament_id = TEMP.tournament_id
            AND GS.player_id = TEMP.player_id;

        -- Drop the temporary table after use
        DROP TABLE #TempPlayerCard;

                            -- Temporary table for #TempFairPlay
        CREATE TABLE #TempFairPlay (
            tournament_id NVARCHAR(100),
            group_id NVARCHAR(100),
                      stage_id INT,
            team_id NVARCHAR(100),
            fair_play_points INT
        );

        -- Insert top scorer (Golden Ball) details
        INSERT INTO #TempFairPlay (tournament_id, group_id, stage_id, team_id,
fair_play_points)
               SELECT
                      TEMP.tournament_id,
                      TEMP.group_id,
                      TEMP.stage_id,
                      TEMP.team_id,
                      SUM(fair_play_points) AS fair_play_points
               FROM (SELECT
                      M.tournament_id,
                      M.group_id,
                      M.stage_id,
                      M.match_id,
                      ME.team_id,
                      SUM(
                             CASE
                                    WHEN B.yellow_card = 1 AND B.second_yellow_card = 0
AND B.red_card = 0 THEN -1
                                    WHEN B.yellow_card = 1 AND B.second_yellow_card = 1
THEN -3
                                    WHEN B.yellow_card = 0 AND B.red_card = 1 THEN -4
                                    WHEN B.yellow_card = 1 AND B.red_card = 1 THEN -5
                                    ELSE 0
                             END
                      ) AS fair_play_points
               FROM
                       WC.[match] AS M
               INNER JOIN
                       WC.[match_event] AS ME ON M.match_id = ME.match_id
               INNER JOIN
                       WC.booking AS B ON ME.event_id = B.event_id
               GROUP BY
                      M.tournament_id,
                      M.group_id,
                      M.stage_id,
                      M.match_id,
                      ME.team_id
               HAVING
                      M.stage_id = 1  ) AS TEMP
               GROUP BY
```

Please read `group_12_trigger_test_cases.pdf`

```
                    TEMP.tournament_id,
                    TEMP.group_id,
                    TEMP.stage_id,
                    TEMP.team_id;


        -- Update group_standing
        UPDATE GS
        SET
            GS.fair_play_points = TEMP.fair_play_points
        FROM
            WC.group_standing AS GS
        INNER JOIN #TempFairPlay AS TEMP
            ON GS.tournament_id = TEMP.tournament_id
            AND GS.group_id = TEMP.group_id
                    AND GS.stage_id = TEMP.stage_id
                    AND GS.team_id = TEMP.team_id;

        DROP TABLE #TempFairPlay;
    END TRY
    BEGIN CATCH
        -- Error handling: Print the error message
        PRINT ERROR_MESSAGE();
    END CATCH
END;
GO
```

Trg_update_team_statistic

Update attributes such ash (wins, draws, losses, performance) after INSERT, UPDATE, DELETE) **team_statistics** table

```
CREATE TRIGGER [WC].[trg_update_team_statistic]
ON  [WC].[team_statistics]
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    -- Use try-catch for error handling
    BEGIN TRY
        -- Declare a temporary table to hold intermediate results
        CREATE TABLE #TempWDL (
            tournament_id NVARCHAR(100),
            team_id NVARCHAR(100),
            wins INT,
            draws INT,
            losses INT
        );

        -- Insert aggregated win, draw, loss data into the temporary table
        INSERT INTO #TempWDL (tournament_id, team_id, wins, draws, losses)
        SELECT TEMP.tournament_id, TEMP.team_id, TEMP.wins, TEMP.draws, TEMP.losses
        FROM
        (
            SELECT
```

Please read `group_12_trigger_test_cases.pdf`

```
                M.tournament_id,
                team.team_id,
                team.team_name,  -- Assuming team_name exists in WC.team
                SUM(CASE
                    WHEN home_team_id = team.team_id AND home_team_win = 1 THEN 1
                    WHEN away_team_id = team.team_id AND away_team_win = 1 THEN 1
                    ELSE 0
                END) AS wins,
                SUM(CASE
                    WHEN (home_team_id = team.team_id OR away_team_id = team.team_id)
AND draw = 1 THEN 1
                    ELSE 0
                END) AS draws,
                SUM(CASE
                    WHEN home_team_id = team.team_id AND away_team_win = 1 THEN 1
                    WHEN away_team_id = team.team_id AND home_team_win = 1 THEN 1
                    ELSE 0
                END) AS losses
            FROM WC.[match] AS M
            JOIN WC.team team ON team.team_id IN (home_team_id, away_team_id)
            GROUP BY M.tournament_id, team.team_id, team.team_name
        ) AS TEMP;

        -- Update the team_statistic table using data from the temporary table
        UPDATE GS
        SET
            GS.wins = TEMP.wins,
            GS.draws = TEMP.draws,
            GS.losses = TEMP.losses
        FROM
             WC.team_statistics AS GS
        INNER JOIN
            #TempWDL AS TEMP ON
            GS.tournament_id = TEMP.tournament_id
            AND GS.team_id = TEMP.team_id;

        -- Drop the temporary table after use
        DROP TABLE #TempWDL;
                    -- Create temporary table for storing performance data
               CREATE TABLE #TempPerformance (
                       tournament_id NVARCHAR(100),
                       team_id NVARCHAR(100),
                       stage_id INT,
                       performance NVARCHAR(100)
               );

               -- Insert performance data into #TempPerformance for new matches only
               INSERT INTO #TempPerformance (tournament_id, team_id, stage_id,
performance)
               SELECT
                       MS.tournament_id,
                       MS.team_id,
                       MS.stage_id,
                       TS.stage_name
               FROM
                       (
```

Please read `group_12_trigger_test_cases.pdf`

```
                        SELECT
                               M.tournament_id,
                               M.team_id,
                               MAX(M.stage_id) AS stage_id
                        FROM (
                               SELECT tournament_id, match_id, stage_id, home_team_id AS
team_id
                               FROM  WC.match
                               UNION ALL
                               SELECT tournament_id, match_id, stage_id, away_team_id AS
team_id
                               FROM  WC.match
                        ) AS M
                        GROUP BY M.tournament_id, M.team_id) AS MS
                INNER JOIN
                         WC.tournament_stage AS TS
                        ON MS.tournament_id = TS.tournament_id
                        AND MS.stage_id = TS.stage_id
                ORDER BY
                        MS.stage_id DESC;

                -- Update team_statistics table with the performance data from
#TempPerformance
                UPDATE GS
                SET GS.performance = TEMP.performance
                FROM WC.team_statistics AS GS
                INNER JOIN #TempPerformance AS TEMP
                        ON GS.tournament_id = TEMP.tournament_id
                        AND GS.team_id = TEMP.team_id
                        AND (GS.performance IS NULL OR GS.performance <> TEMP.performance);

                -- Clean up the temporary table
                DROP TABLE #TempPerformance
    END TRY
    BEGIN CATCH
        -- Error handling: Print the error message
        PRINT ERROR_MESSAGE();
    END CATCH
END;
GO
```

Trg_update_goals_stats

Update all kind of goal attributes for tables:

- Group_standing (goals_for, goals_against)
- Team_statistics (goals_for, goals_against)
- Player_statistics (goals_for)

```
CREATE TRIGGER [WC].[trg_update_goals_stats]
ON  [WC].[goal]
AFTER INSERT, UPDATE, DELETE
AS
```

Please read `group_12_trigger_test_cases.pdf`

```
BEGIN
    -- Declare a temporary table to hold intermediate results
    CREATE TABLE #TempGroupGoals (
        tournament_id NVARCHAR(100),
        team_id NVARCHAR(100),
        goals_for INT,
        goals_against INT
    );

       -- Declare a temporary table to hold intermediate results
    CREATE TABLE #TempTeamGoals (
        tournament_id NVARCHAR(100),
        team_id NVARCHAR(100),
        goals_for INT,
        goals_against INT
    );

        -- Declare a temporary table to hold intermediate results
    CREATE TABLE #TempPlayerGoals (
        tournament_id NVARCHAR(100),
        player_id NVARCHAR(100),
        goals_for INT
    );

    -- Insert aggregated goal data into the temporary table
    INSERT INTO #TempGroupGoals (tournament_id, team_id, goals_for, goals_against)
       SELECT
              TEMP.tournament_id,
              TEMP.team_id,
              SUM(TEMP.goals_for),
              SUM(TEMP.goals_against)
       FROM ( SELECT
                           GS.tournament_id,
                           GS.team_id,
                           GOAL.match_id,
                           SUM(CASE
                                  WHEN GOAL.own_goal = 0 AND GOAL.team_id =
GOAL.player_team_id THEN 1   -- Team scores
                                  WHEN GOAL.own_goal = 1 AND GOAL.team_id <>
GOAL.player_team_id THEN 1   -- Opponent scores due to own goal
                                  ELSE 0
                           END) AS goals_for,
                           SUM(CASE
                                  WHEN GOAL.own_goal = 0 AND GOAL.team_id <>
GOAL.player_team_id THEN 1   -- Opponent scores
                                  WHEN GOAL.own_goal = 1 AND GOAL.team_id =
GOAL.player_team_id THEN 1   -- Own goal
                                  ELSE 0
                           END) AS goals_against
                     FROM
                           (SELECT
                                  M.match_id,
                                  M.tournament_id,
                                  M.home_team_id AS team_id,
                                  ME.player_id,
                                  ME.team_id AS 'player_team_id',
```

Please read `group_12_trigger_test_cases.pdf`

```
                                G.own_goal
                        FROM
                                WC.match AS M
                        INNER JOIN
                                WC.match_event AS ME ON M.match_id = ME.match_id
                        INNER JOIN
                                WC.goal AS G ON ME.event_id = G.event_id

                        UNION ALL

                        SELECT
                                M.match_id,
                                M.tournament_id,
                                M.away_team_id AS team_id,
                                ME.player_id,
                                ME.team_id AS 'player_team_id',
                                G.own_goal
                        FROM
                                WC.match AS M
                        INNER JOIN
                                WC.match_event AS ME ON M.match_id = ME.match_id
                        INNER JOIN
                                WC.goal AS G ON ME.event_id = G.event_id) AS GOAL
                INNER JOIN
                         WC.group_standing AS GS ON
                        GOAL.tournament_id = GS.tournament_id
                        AND GOAL.team_id = GS.team_id
                GROUP BY
                        GS.tournament_id,
                        GS.team_id,
                        GOAL.match_id
                HAVING CAST(SUBSTRING(GOAL.match_id, 8, 9) AS int) < 49 ) AS TEMP
        GROUP BY TEMP.tournament_id, TEMP.team_id;

    -- Update the group_standing table using data from the temporary table
    UPDATE GS
    SET
        GS.goals_for = COALESCE(TEMP.goals_for, 0),
        GS.goals_against = COALESCE(TEMP.goals_against, 0)
              --, GS.goals_difference = COALESCE(TEMP.goals_for, 0) -
COALESCE(TEMP.goals_against, 0)
    FROM
         WC.group_standing AS GS
    INNER JOIN
        #TempGroupGoals AS TEMP ON
        GS.tournament_id = TEMP.tournament_id
        AND GS.team_id = TEMP.team_id;

    -- Drop the temporary table after use
    DROP TABLE #TempGroupGoals;

    -- Insert aggregated goal data into the temporary table
    INSERT INTO #TempTeamGoals (tournament_id, team_id, goals_for, goals_against)
       SELECT
              TEMP.tournament_id,
              TEMP.team_id,
```

Please read `group_12_trigger_test_cases.pdf`

```
            SUM(TEMP.goals_for),
            SUM(TEMP.goals_against)
       FROM ( SELECT
                           GS.tournament_id,
                           GS.team_id,
                           GOAL.match_id,
                           SUM(CASE
                                  WHEN GOAL.own_goal = 0 AND GOAL.team_id =
GOAL.player_team_id THEN 1   -- Team scores
                                  WHEN GOAL.own_goal = 1 AND GOAL.team_id <>
GOAL.player_team_id THEN 1   -- Opponent scores due to own goal
                                  ELSE 0
                           END) AS goals_for,
                           SUM(CASE
                                  WHEN GOAL.own_goal = 0 AND GOAL.team_id <>
GOAL.player_team_id THEN 1   -- Opponent scores
                                  WHEN GOAL.own_goal = 1 AND GOAL.team_id =
GOAL.player_team_id THEN 1   -- Own goal
                                  ELSE 0
                           END) AS goals_against
                     FROM
                           (SELECT
                                  M.match_id,
                                  M.tournament_id,
                                  M.home_team_id AS team_id,
                                  ME.player_id,
                                  ME.team_id AS 'player_team_id',
                                  G.own_goal
                           FROM
                                   WC.match AS M
                           INNER JOIN
                                   WC.match_event AS ME ON M.match_id = ME.match_id
                           INNER JOIN
                                   WC.goal AS G ON ME.event_id = G.event_id

                           UNION ALL

                           SELECT
                                  M.match_id,
                                  M.tournament_id,
                                  M.away_team_id AS team_id,
                                  ME.player_id,
                                  ME.team_id AS 'player_team_id',
                                  G.own_goal
                           FROM
                                   WC.match AS M
                           INNER JOIN
                                   WC.match_event AS ME ON M.match_id = ME.match_id
                           INNER JOIN
                                   WC.goal AS G ON ME.event_id = G.event_id) AS GOAL
                     INNER JOIN
                            WC.team_statistics AS GS ON
                           GOAL.tournament_id = GS.tournament_id
                           AND GOAL.team_id = GS.team_id
                     GROUP BY
                           GS.tournament_id,
```

Please read `group_12_trigger_test_cases.pdf`

```
                        GS.team_id,
                        GOAL.match_id ) AS TEMP
            GROUP BY TEMP.tournament_id, TEMP.team_id;

    -- Update the team_statistic table using data from the temporary table
    UPDATE GS
    SET
        GS.goals = COALESCE(TEMP.goals_for, 0),
        GS.goals_against = COALESCE(TEMP.goals_against, 0)
    FROM
         WC.team_statistics AS GS
    INNER JOIN
        #TempTeamGoals AS TEMP ON
        GS.tournament_id = TEMP.tournament_id
        AND GS.team_id = TEMP.team_id;

    -- Drop the temporary table after use
    DROP TABLE #TempTeamGoals;

    -- Insert aggregated goal data into the temporary table
    INSERT INTO #TempPlayerGoals (tournament_id, player_id, goals_for)
       SELECT
       TEMP.tournament_id,
       TEMP.player_id,
       SUM(TEMP.goal)
       FROM
              (SELECT
                     M.tournament_id,
                     M.match_id,
                     ME.player_id,
                     SUM(CASE WHEN G.own_goal = 0 THEN 1 ELSE 0 END) AS 'goal'
              FROM
                      WC.match AS M
              INNER JOIN
                      WC.match_event AS ME
              ON
                     M.match_id = ME.match_id
              INNER JOIN
                      WC.goal AS G
              ON
                     ME.event_id = G.event_id
              GROUP BY
                     M.tournament_id,
                     M.match_id,
                     ME.player_id ) AS TEMP
       GROUP BY
              TEMP.tournament_id,
              TEMP.player_id;

    -- Update the team_statistic table using data from the temporary table
    UPDATE GS
    SET
        GS.goals = COALESCE(TEMP.goals_for, 0)
    FROM
         WC.player_statistics AS GS
    INNER JOIN
```

Please read `group_12_trigger_test_cases.pdf`

```
        #TempPlayerGoals AS TEMP ON
        GS.tournament_id = TEMP.tournament_id
        AND GS.player_id = TEMP.player_id;

    -- Drop the temporary table after use
    DROP TABLE #TempPlayerGoals;
END;
GO
```

Trg_insert_tournament_standing

After all matches of the tournament happened, we know which team is the winner, 2nd-place, 3rd-place, and 4th-place.
As we have **stage_id** from the **tournament_stage** table, we will know:

- Stage_id = 5: Third place match
- Stage_id = 6: Final match

From that, we can find the top 4 of that tournament.

```
CREATE TRIGGER [WC].[trg_insert_tournament_standing]
ON [WC].[match]
AFTER INSERT, UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    -- Use MERGE to insert or update tournament_standing based on the results of matches
    MERGE INTO WC.tournament_standing AS target
    USING (
        SELECT
            tournament_id,
            CASE WHEN home_team_win = 1 THEN home_team_id ELSE away_team_id END AS
team_id,
            CASE
                WHEN stage_id = 6 THEN 1  -- Winner of the final
                WHEN stage_id = 5 THEN 3  -- Winner of the third-place match
            END AS position
        FROM WC.[match]
        WHERE stage_id IN (5, 6)
              UNION
              SELECT
            tournament_id,
            CASE WHEN home_team_win = 1 THEN away_team_id ELSE home_team_id END AS
team_id,
            CASE
                WHEN stage_id = 6 THEN 2  -- Winner of the final
                WHEN stage_id = 5 THEN 4  -- Winner of the third-place match
            END AS position
```

Please read `group_12_trigger_test_cases.pdf`

```
        FROM WC.[match]
        WHERE stage_id IN (5, 6)
    ) AS source
    ON target.tournament_id = source.tournament_id AND target.team_id = source.team_id

    WHEN MATCHED THEN
        UPDATE SET target.position = source.position  -- Update position if the record
exists
    WHEN NOT MATCHED THEN
        INSERT (tournament_id, team_id, position)
        VALUES (source.tournament_id, source.team_id, source.position);  -- Insert new
record

END;
GO
```

Trg_after_tournament_insert_tournament_statistics

After insert data into **tournament** table, also insert (**tournament_id**) into **tournament_statistics** table if not exists.

```
CREATE TRIGGER [WC].[trg_after_tournament_insert_tournament_statistics]
ON  [WC].[tournament]
AFTER INSERT, UPDATE
AS
BEGIN
       -- Insert into team_statistic table
       INSERT INTO  WC.tournament_statistics(tournament_id)
    SELECT i.tournament_id
    FROM inserted i
    WHERE NOT EXISTS (
        SELECT 1
        FROM  WC.tournament_statistics ts
        WHERE ts.tournament_id = i.tournament_id
    );
       -- Update the tournament_statistics table
       UPDATE ts
    SET ts.tournament_id = i.tournament_id
    FROM WC.tournament_statistics AS ts
    JOIN inserted i
    ON ts.tournament_id = i.tournament_id;
END;
GO
```

Trg_update_tournament_summary

- After insert and update all records from **team_statistics** and **player_statistics**. Aggregate to update attributes (count_matches, count_teams, yellow_cards, red_cards, sent_offs)

Please read `group_12_trigger_test_cases.pdf`

```
CREATE TRIGGER [WC].[trg_update_tournament_summary]
ON [WC].[player_statistics]
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- Update count_matches in tournament_summary
    UPDATE WC.tournament_statistics
    SET count_matches = M.count_matches
    FROM (
        SELECT
            tournament_id,
            COUNT(match_id) AS count_matches
        FROM
            WC.match
        GROUP BY
            tournament_id
    ) AS M
    WHERE WC.tournament_statistics.tournament_id = M.tournament_id;

    -- Update count_teams in tournament_summary
    UPDATE WC.tournament_statistics
    SET count_teams = T.count_teams
    FROM (
        SELECT
            tournament_id,
            COUNT(DISTINCT team_id) AS count_teams
        FROM
            WC.squad
        GROUP BY
            tournament_id
    ) AS T
    WHERE WC.tournament_statistics.tournament_id = T.tournament_id;

    -- Update goals, yellow_cards, and red_cards in tournament_summary
    UPDATE WC.tournament_statistics
    SET
        goals = TS.goals,
        yellow_cards = TS.yellow_cards,
        red_cards = TS.red_cards
    FROM (
        SELECT
            tournament_id,
            SUM(goals) AS goals,
            SUM(yellow_cards) AS yellow_cards,
            SUM(red_cards) AS red_cards
        FROM
            WC.team_statistics
        GROUP BY
            tournament_id
    ) AS TS
    WHERE WC.tournament_statistics.tournament_id = TS.tournament_id;

    -- Update sent_off in tournament_summary
    UPDATE WC.tournament_statistics
```

Please read `group_12_trigger_test_cases.pdf`

```
    SET sent_offs = PS.sent_off
    FROM (
        SELECT
            tournament_id,
            SUM(sent_offs) AS sent_off
        FROM
            WC.player_statistics
        GROUP BY
            tournament_id
    ) AS PS
    WHERE WC.tournament_statistics.tournament_id = PS.tournament_id;

END;
GO
```

EXPLAIN THE TRIGGERS TO GENERATE GROUP CLASSIFICATION

We create triggers to insert data into group_standing table attributes (tournament_id, group_id, stage_id, team_id)

After that, we create multiple triggers to UPDATE data from **match_event** tables to other **group_standing** attributes

Trg_update_group_standing

```
CREATE TRIGGER [WC].[trg_update_group_standing]
ON  [WC].[group_standing]
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
       BEGIN TRY
              -- Declare a temporary table to hold intermediate results
              CREATE TABLE #TempGoals (
                  tournament_id NVARCHAR(100),
                  team_id NVARCHAR(100),
                  played INT
              );
              -- Insert aggregated goal data into the temporary table
              INSERT INTO #TempGoals (tournament_id, team_id, played)
              SELECT
                     TEMP.tournament_id
                     , TEMP.team_id
                     , COUNT(TEMP.match_id) AS played
```

Please read `group_12_trigger_test_cases.pdf`

```
                FROM
                        (SELECT
                                tournament_id, home_team_id AS team_id, match_id, stage_id
                        FROM  WC.match
                        WHERE stage_id = 1
                        UNION
                        SELECT tournament_id, away_team_id AS team_id, match_id, stage_id
                        FROM  WC.match
                        WHERE stage_id = 1 ) AS TEMP
                GROUP BY
                        TEMP.tournament_id,
                        TEMP.team_id;

                -- Update the team_statistic table using data from the temporary table
                UPDATE GS
                SET
                    GS.played = COALESCE(TEMP.played, 0)
                FROM
                     WC.group_standing AS GS
                INNER JOIN
                    #TempGoals AS TEMP ON
                    GS.tournament_id = TEMP.tournament_id
                    AND GS.team_id = TEMP.team_id;

                -- Drop the temporary table after use
                DROP TABLE #TempGoals;

                -- Declare a temporary table to hold intermediate results
                CREATE TABLE #TempWDL (
                    tournament_id NVARCHAR(100),
                    team_id NVARCHAR(100),
                    wins INT,
                        draws INT,
                        losses INT
                );
                -- Insert aggregated goal data into the temporary table
                INSERT INTO #TempWDL (tournament_id, team_id, wins, draws, losses)
                SELECT TEMP.tournament_id, TEMP.team_id, TEMP.wins, TEMP.draws,
TEMP.losses
                FROM
                (       SELECT
                        M.tournament_id,
                        M.stage_id,
                        team.team_id,
                        team.team_name,  -- Assuming team_name exists in WC.team
                        SUM(CASE
                                WHEN home_team_id = team.team_id AND home_team_win = 1 THEN
1
                                WHEN away_team_id = team.team_id AND away_team_win = 1 THEN
1
                                ELSE 0
                        END) AS wins,
                        SUM(CASE
                                WHEN (home_team_id = team.team_id OR away_team_id =
team.team_id) AND draw = 1 THEN 1
                                ELSE 0
```

Please read `group_12_trigger_test_cases.pdf`

```
                    END) AS draws,
                    SUM(CASE
                            WHEN home_team_id = team.team_id AND away_team_win = 1 THEN
1
                            WHEN away_team_id = team.team_id AND home_team_win = 1 THEN
1
                            ELSE 0
                    END) AS losses

            FROM WC.[match] AS M
            JOIN WC.team team ON team.team_id IN (home_team_id, away_team_id)
            GROUP BY M.tournament_id, M.stage_id, team.team_id, team.team_name
            HAVING M.stage_id = 1) AS TEMP

            -- Update the team_statistic table using data from the temporary table
            UPDATE GS
            SET
                GS.wins = TEMP.wins,
                GS.draws = TEMP.draws,
                GS.losses = TEMP.losses
            FROM
                 WC.group_standing AS GS
            INNER JOIN
                #TempWDL AS TEMP ON
                GS.tournament_id = TEMP.tournament_id
                AND GS.team_id = TEMP.team_id;

            -- Drop the temporary table after use
            DROP TABLE #TempWDL;



        -- Temporary table for #TempGroupStandingPosition
        CREATE TABLE #TempGroupStandingPosition (
            tournament_id NVARCHAR(100),
            group_id NVARCHAR(100),
            team_id NVARCHAR(100),
            posititon INT,
                    advanced BIT
        );

        -- Insert top scorer (Golden Ball) details
        INSERT INTO #TempGroupStandingPosition (tournament_id, group_id, team_id,
posititon, advanced)
            SELECT
                    tournament_id,
                    group_id,
                    team_id,
                    position,
                    CASE WHEN position <= 2 THEN 1 ELSE 0 END AS advanced
            FROM (
                    SELECT
                            tournament_id,
                            group_id,
                            team_id,
                            points,
```

Please read `group_12_trigger_test_cases.pdf`

```
                            goals_difference,
                            goals_for,
                            ROW_NUMBER() OVER (
                                    PARTITION BY tournament_id, group_id
                                    ORDER BY points DESC, goals_difference DESC,
goals_for DESC, fair_play_points DESC
                            ) AS position
                        FROM
                                    WC.group_standing
                ) AS TEMP


        -- Update group_standing
        UPDATE GS
        SET
            GS.position = TEMP.posititon,
            GS.advanced = TEMP.advanced
        FROM
            WC.group_standing AS GS
        INNER JOIN #TempGroupStandingPosition AS TEMP
            ON GS.tournament_id = TEMP.tournament_id
            AND GS.group_id = TEMP.group_id
                        AND GS.team_id = TEMP.team_id;

        DROP TABLE #TempGroupStandingPosition;
    END TRY
    BEGIN CATCH
        PRINT ERROR_MESSAGE();
    END CATCH
END;
GO
```

EXPLAIN THE TRIGGER TO GENERATE AWARD WINNER

Firstly, we have a trigger to insert data into the **award_winner** table (tournament_id, award_id)

Trg_insert_award_winner

When inserting a new award into the **award** table, insert a new record for the **award_winner** table if not exist.

```
CREATE TRIGGER [WC].[trg_insert_award_winner]
ON [WC].[award] -- Replace with your target table
AFTER INSERT
AS
```

Please read `group_12_trigger_test_cases.pdf`

```
BEGIN
    SET NOCOUNT ON;

    -- Insert only if the record does not already exist
    INSERT INTO WC.award_winner (tournament_id, award_id)
    SELECT
        T.tournament_id,
        i.award_id
    FROM inserted i
       CROSS JOIN  WC.tournament AS T
    WHERE NOT EXISTS (
        SELECT 1
        FROM WC.award_winner aw
        WHERE aw.tournament_id = T.tournament_id
        AND aw.award_id = i.award_id
    );
END;
GO
```

Then, we create 2 triggers:

- Update Golden Ball, Silver Boot, Bronze Boot
- Update Best Goal Keeper

Trg_update_award_winner_golden_sivler_bronze_boot

From **player_statistics** table, select top 3 best player to achieve (Golden Ball, Silver Boot, Bronze Boot)

```
CREATE TRIGGER [WC].[trg_update_award_winner_golden_sivler_bronze_boot]
ON [WC].[player_statistics]
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY
        -- Temporary table for Golden Ball
        CREATE TABLE #TempGoldenSilverBronze (
            tournament_id NVARCHAR(100),
            award_id NVARCHAR(100),
            team_id NVARCHAR(100),
            player_id NVARCHAR(100)
        );

        -- Insert top scorer (Golden Ball) details
        INSERT INTO #TempGoldenSilverBronze (tournament_id, award_id, team_id,
player_id)
        SELECT
            ts.tournament_id,
            A.award_id,
            S.team_id,
            ts.player_id
```

Please read `group_12_trigger_test_cases.pdf`

```
        FROM
            (SELECT
                ps.tournament_id,
                ps.player_id,
                ROW_NUMBER() OVER (PARTITION BY ps.tournament_id ORDER BY ps.goals DESC)
AS GoalRank
             FROM
                WC.player_statistics ps
            ) ts
        INNER JOIN WC.award AS A
            ON A.award_name = 'Golden Ball'
        INNER JOIN WC.squad AS S
            ON ts.tournament_id = S.tournament_id AND ts.player_id = S.player_id
        WHERE ts.GoalRank = 1;

              -- Insert 2nd top scorer (Silver Ball) details
        INSERT INTO #TempGoldenSilverBronze (tournament_id, award_id, team_id,
player_id)
        SELECT
            ts.tournament_id,
            A.award_id,
            S.team_id,
            ts.player_id
        FROM
            (SELECT
                ps.tournament_id,
                ps.player_id,
                ROW_NUMBER() OVER (PARTITION BY ps.tournament_id ORDER BY ps.goals DESC)
AS GoalRank
             FROM
                WC.player_statistics ps
            ) ts
        INNER JOIN WC.award AS A
            ON A.award_name = 'Silver Boot'
        INNER JOIN WC.squad AS S
            ON ts.tournament_id = S.tournament_id AND ts.player_id = S.player_id
        WHERE ts.GoalRank = 2;

              -- Insert 3rd top scorer (Bronze Boot) details
        INSERT INTO #TempGoldenSilverBronze (tournament_id, award_id, team_id,
player_id)
        SELECT
            ts.tournament_id,
            A.award_id,
            S.team_id,
            ts.player_id
        FROM
            (SELECT
                ps.tournament_id,
                ps.player_id,
                ROW_NUMBER() OVER (PARTITION BY ps.tournament_id ORDER BY ps.goals DESC)
AS GoalRank
             FROM
                WC.player_statistics ps
            ) ts
        INNER JOIN WC.award AS A
```

Please read `group_12_trigger_test_cases.pdf`

```
            ON A.award_name = 'Bronze Boot'
        INNER JOIN WC.squad AS S
            ON ts.tournament_id = S.tournament_id AND ts.player_id = S.player_id
        WHERE ts.GoalRank = 3;

        -- Update award_winner table for Golden Ball
        UPDATE AW
        SET
            AW.team_id = TEMP.team_id,
            AW.player_id = TEMP.player_id
        FROM
            WC.award_winner AS AW
        INNER JOIN #TempGoldenSilverBronze AS TEMP
            ON AW.tournament_id = TEMP.tournament_id
            AND AW.award_id = TEMP.award_id;

        DROP TABLE #TempGoldenSilverBronze;
    END TRY
    BEGIN CATCH
        PRINT ERROR_MESSAGE();
    END CATCH
END
GO
```

Trg_update_best_goalkeeper

To find best goalkeeper of each tournaments means found **Goalkeeper (Position_id: GK)** who concedes the least `MIN(concedes)` and played the most matches during that tournament `MAX(matches)`

```
CREATE   TRIGGER [WC].[trg_update_best_goalkeeper]
ON  [WC].[match_event]
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    -- Step 1: Calculate matches played and goals conceded for each goalkeeper
    DROP TABLE IF EXISTS #tmp_test1, #tmp_test2;

    SELECT
        M.tournament_id,
        PA.player_id,
        COUNT(DISTINCT M.match_id) AS matches, -- Count distinct matches played by the
goalkeeper
        SUM(CASE
            WHEN ME.team_id = S.team_id AND G.own_goal = 0 THEN 0 -- Non-own goal by the
goalkeeper's team
            WHEN ME.team_id = S.team_id AND G.own_goal = 1 THEN 1 -- Own goal by the
goalkeeper's team
            WHEN ME.team_id <> S.team_id AND G.own_goal = 0 THEN 1 -- Non-own goal by
opponent (conceded)
            WHEN ME.team_id <> S.team_id AND G.own_goal = 1 THEN 0 -- Own goal by
opponent (not counted against)
        END) AS concedes -- Total goals conceded by the goalkeeper
```

Please read `group_12_trigger_test_cases.pdf`

```
    INTO #tmp_test1
    FROM
         WC.match_event AS ME
    INNER JOIN
         WC.[match] AS M ON ME.match_id = M.match_id
    INNER JOIN
         WC.goal AS G ON ME.event_id = G.event_id
    INNER JOIN
         WC.player_appearance AS PA ON PA.match_id = M.match_id
    INNER JOIN
         WC.squad AS S ON PA.player_id = S.player_id AND M.tournament_id =
S.tournament_id
    WHERE
        PA.position_id = 'GK' -- Filtering for goalkeepers
    GROUP BY
        M.tournament_id,
        PA.player_id;

    -- Step 2: Find max matches for each concede level
    WITH MAX_MATCH AS (
        SELECT
            tournament_id,
            concedes,
            MAX(matches) AS max_matches
        FROM
            #tmp_test1
        GROUP BY
            tournament_id,
            concedes
    )
    SELECT
        tournament_id,
        max_matches,
        MIN(concedes) AS min_concedes
    INTO #tmp_test2
    FROM
        MAX_MATCH
    GROUP BY
        tournament_id,
        max_matches;

    -- Step 3: Update award_winner table with the goalkeepers having max matches and min
concedes
    DELETE FROM  WC.award_winner
    WHERE award_id = 'A-7'; -- Remove previous entries for the clean sheet award

    INSERT INTO  WC.award_winner (tournament_id, team_id, player_id, award_id)
    SELECT
        CLEAN_SHEET.tournament_id,
        S.team_id,
        CLEAN_SHEET.player_id,
        'A-7' -- Clean sheet award ID
    FROM
        (SELECT
            t1.*,
            ROW_NUMBER() OVER(PARTITION BY t1.tournament_id ORDER BY t2.max_matches
```

Please read `group_12_trigger_test_cases.pdf`

```
DESC) AS RN
         FROM
            #tmp_test1 AS t1
         INNER JOIN
            #tmp_test2 AS t2 ON t1.matches = t2.max_matches
            AND t1.concedes = t2.min_concedes
        ) AS CLEAN_SHEET
    INNER JOIN
         WC.squad AS S ON CLEAN_SHEET.tournament_id = S.tournament_id AND
CLEAN_SHEET.player_id = S.player_id
    WHERE
        CLEAN_SHEET.RN = 1; -- Only insert the top goalkeeper per tournament

    -- Cleanup temporary tables
    DROP TABLE IF EXISTS #tmp_test1, #tmp_test2;
END;
GO
```