```sql
DELIMITER $$

DROP PROCEDURE IF EXISTS ApplyDiscountByCriteria $$

CREATE PROCEDURE ApplyDiscountByCriteria(
    IN p_genre VARCHAR(255),
    IN p_release_date VARCHAR(255),
    IN p_game_name VARCHAR(255),
    IN p_discount_rate DECIMAL(5,2))
BEGIN
    -- Declare necessary variables and the cursor
    DECLARE done INT DEFAULT FALSE;
    DECLARE v_gameName VARCHAR(255);
    DECLARE v_oldPrice DECIMAL(10,4);
    DECLARE v_oldFinalPrice DECIMAL(10,4);
    DECLARE v_newPrice DECIMAL(10,4);

    DECLARE gameCursor CURSOR FOR
        -- do not allow making a discount on bookmarked games, so we can handle that separately later
        SELECT g.gameName, g.initialPrice
        FROM Games AS g JOIN GameGenre AS gg ON g.gameName = gg.gameName
        WHERE (gg.gameGenre = p_genre OR p_genre = '')
            AND (g.releaseDate = p_release_date OR p_release_date = '')
            AND (g.gameName LIKE CONCAT('%', p_game_name, '%'))
            AND g.gameName NOT IN (SELECT g2.gameName
                                   FROM Games g2 NATURAL JOIN Bookmarks b);

    DECLARE bookmarksCursor CURSOR FOR
        -- now handle bookmarks separately
        SELECT g.gameName, g.initialPrice, g.finalPrice
        FROM Games AS g JOIN GameGenre AS gg ON g.gameName = gg.gameName
        WHERE (gg.gameGenre = p_genre OR p_genre = '')
            AND (g.releaseDate = p_release_date OR p_release_date = '')
            AND (g.gameName LIKE CONCAT('%', p_game_name, '%'))
            AND g.gameName IN (SELECT g2.gameName
                               FROM Games g2 NATURAL JOIN Bookmarks b);

    -- Declare a continue handler for NOT FOUND condition
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- Open the cursor
    OPEN gameCursor;

    -- Start the loop
    read_loop: LOOP
        -- Fetch the next game from the cursor
        FETCH gameCursor INTO v_gameName, v_oldPrice;

        -- Check if the cursor reached the end of fetched records
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- Calculate the new price after applying the discount
        SET v_newPrice = v_oldPrice * (1 - p_discount_rate);

        -- Update the finalPrice in the Games table
        UPDATE Games
        SET finalPrice = v_newPrice
        WHERE gameName = v_gameName;

        -- Continue to the next iteration of the loop
    END LOOP;

    -- Close the cursor
    CLOSE gameCursor;

    SET done = false;

    -- Open the cursor
    OPEN bookmarksCursor;

    -- Start the loop
    read_loop_two: LOOP
        -- Fetch the next game from the cursor
        FETCH bookmarksCursor INTO v_gameName, v_oldPrice, v_oldFinalPrice;

        -- Check if the cursor reached the end of fetched records
        IF done THEN
            LEAVE read_loop_two;
        END IF;

        -- Calculate the new price after applying the discount
        SET v_newPrice = v_oldPrice * (1 - p_discount_rate);

        -- Only apply discount to bookmarked games when cheaper
        IF v_newPrice < v_oldFinalPrice THEN
            -- Update the finalPrice in the Games table
            UPDATE Games
            SET finalPrice = v_newPrice
            WHERE gameName = v_gameName;
        END IF;
        -- Continue to the next iteration of the loop
    END LOOP;

    -- Close the cursor
    CLOSE bookmarksCursor;

    -- After closing the cursor, SELECT the updated records, along with lowest min ever price
    SELECT g.gameName, gg.gameGenre, g.releaseDate, pl.old_price, pl.new_price, (SELECT MIN(pl2.new_price) FROM price_change_log pl2 WHERE pl.gameName = pl2.gameName GROUP BY pl2.gameName) as lowest_price
    FROM price_change_log AS pl JOIN Games AS g ON pl.gameName = g.gameName JOIN GameGenre AS gg ON g.gameName = gg.gameName
    WHERE pl.changed_at >= NOW() - INTERVAL 1 SECOND;
END $$
```

```
CREATE TRIGGER log_price_change
AFTER UPDATE ON Games
FOR EACH ROW
BEGIN
    IF NEW.finalPrice <> OLD.finalPrice THEN
        INSERT INTO price_change_log (gameName, old_price, new_price, changed_at)
        VALUES (OLD.gameName, OLD.finalPrice, NEW.finalPrice, NOW());
    END IF;
END$$

DELIMITER ;

```