Constraints:

```
ALTER TABLE UserRecommendedPerfumes DROP FOREIGN KEY
UserRecommendedPerfumes_ibfk_1;

ALTER TABLE UserRecommendedPerfumes ADD CONSTRAINT
UserRecommendedPerfumes_ibfk_1 FOREIGN KEY (UserID) REFERENCES User(UserID)
ON DELETE CASCADE;
```

Triggers:

1) Everytime a user's notes & feelings are updated (retaking the quiz), update the UserRecommendedPerfumes with top matches (determined by MatchScore) based on these new values

```
DELIMITER //

CREATE TRIGGER AfterUserUpdate
AFTER UPDATE ON User
FOR EACH ROW
BEGIN
IF NEW.Notes != OLD.Notes OR NEW.Feelings != OLD.Feelings THEN
DELETE FROM UserRecommendedPerfumes WHERE UserID = NEW.UserID;

INSERT INTO UserRecommendedPerfumes (UserID, PerfumeID, Name, Brand, Price, Image,
MatchScore)
SELECT
NEW.UserID,
p.PerfumeID,
p.Name,
p.Brand,
p.Price,
p.Image,
(

(
LENGTH(NEW.Notes) - LENGTH(REPLACE(NEW.Notes, s.Notes, '')) +
LENGTH(NEW.Feelings) - LENGTH(REPLACE(NEW.Feelings, s.Feelings, ''))
)
/

(
LENGTH(NEW.Notes) - LENGTH(REPLACE(NEW.Notes, ',', '')) + 1 +
LENGTH(NEW.Feelings) - LENGTH(REPLACE(NEW.Feelings, ',', '')) + 1
)
) * 100 AS MatchScore
```

```
FROM
Perfumes p
JOIN
Scent s ON p.PerfumeID = s.PerfumeID
WHERE
LENGTH(NEW.Notes) > 0 AND LENGTH(NEW.Feelings) > 0
ORDER BY
MatchScore DESC
LIMIT 60;
END IF;
END//

DELIMITER ;
```

Transactions:

1) Getting the top notes from top reviewers as inspiration for what to search for

```
DELIMITER //

CREATE PROCEDURE AnalyzeTopNotes()
BEGIN
DECLARE maxReviewCount INT;

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
ROLLBACK;

START TRANSACTION;

WITH TopReviewers AS (
SELECT ReviewerName
FROM Reviews
GROUP BY ReviewerName
ORDER BY COUNT(*) DESC
LIMIT 2
)
SELECT r.Notes AS Note, COUNT(*) AS NoteCount
FROM Reviews r
JOIN TopReviewers tr ON r.ReviewerName = tr.ReviewerName
GROUP BY r.Notes
ORDER BY NoteCount DESC
LIMIT 1;

UPDATE Reviews
SET ReviewerName = CONCAT(ReviewerName, '*')
WHERE ReviewerName IN (SELECT ReviewerName FROM TopReviewers);
```

```
        COMMIT;

        END //

        DELIMITER ;
```

Stored Procedures:

1) Get dupes for specific on each of the search bar results, which we use to show the dupes of the perfumes given as results of the quiz

```
DELIMITER $$

CREATE PROCEDURE getDupesForOne(IN perfumeID VARCHAR(8))
BEGIN
SELECT
d.DupeID AS DupeID,
s1.PerfumeID AS OriginalPerfumeID,
p1.Name AS OriginalPerfumeName, p1.Price AS Price1,
s1.Notes AS OriginalNotes,
s1.Feelings AS OriginalFeelings,
s2.PerfumeID AS DupePerfumeID,
p2.Name AS DupePerfumeName, p2.Price AS Price2,
s2.Notes AS DupeNotes,
s2.Feelings AS DupeFeelings
FROM Scent s1
JOIN Dupes d ON s1.PerfumeID = d.OriginalID
JOIN Scent s2 ON s2.PerfumeID = d.DupeID
JOIN Perfumes p1 ON p1.PerfumeID = s1.PerfumeID
JOIN Perfumes p2 ON p2.PerfumeID = s2.PerfumeID
WHERE d.OriginalID = perfumeID
AND (s2.Notes = s1.Notes OR s2.Feelings = s1.Feelings)

UNION

SELECT
d.OriginalID AS DupeID,
s2.PerfumeID AS OriginalPerfumeID,
p2.Name AS OriginalPerfumeName, p2.Price AS Price2,
s2.Notes AS OriginalNotes,
s2.Feelings AS OriginalFeelings,
s1.PerfumeID AS DupePerfumeID,
p1.Name AS DupePerfumeName, p1.Price AS Price1,
s1.Notes AS DupeNotes,
s1.Feelings AS DupeFeelings
FROM Scent s2
JOIN Dupes d ON s2.PerfumeID = d.DupeID
JOIN Scent s1 ON s1.PerfumeID = d.OriginalID
JOIN Perfumes p1 ON p1.PerfumeID = s1.PerfumeID
JOIN Perfumes p2 ON p2.PerfumeID = s2.PerfumeID
WHERE d.DupeID = perfumeID
AND (s1.Notes = s2.Notes OR s1.Feelings = s2.Feelings);
END$$

DELIMITER ;
```

2) Getting Avg dupe price for each perfume shown on the results of the quiz page, to allow users to see how much of a price difference they can get.

```
DELIMITER $$

CREATE PROCEDURE GetAvgDupePrice(IN PerfumeID INT)
BEGIN
SELECT p.Name AS PerfumeName, p.Brand,
AVG(CAST(d_perf.Price AS DECIMAL(10, 2))) AS AvgDupePrice
FROM Perfumes p
JOIN Dupes d ON p.PerfumeID = d.OriginalID
JOIN Perfumes d_perf ON d.DupeID = d_perf.PerfumeID
WHERE p.PerfumeID = PerfumeID -- Use the parameter directly
GROUP BY p.PerfumeID, p.Name, p.Brand;
END$$

DELIMITER ;
```