HW 3. SQL - Solution

Assignment: <http://goo.gl/0I5quH>.

USE ReviewApplication;

# 1. What are the top 10 average review ratings for restaurants? Include the restaurant ID, restaurant name, and average rating in the result set.

# Note the usage of fully qualified column names. This isn't necessary in MySQL

# if the column names are unique across the referenced tables.

SELECT Restaurants.RestaurantId, Restaurants.Name, AVG(Reviews.Rating) AS AvgRtg

FROM Restaurants

INNER JOIN Reviews

ON Restaurants.RestaurantId = Reviews.RestaurantId

GROUP BY Restaurants.RestaurantId, Restaurants.Name

ORDER BY AvgRtg DESC, Restaurants.RestaurantId ASC, Restaurants.Name ASC

LIMIT 10;

# 2. How many users have created more than one review?

# Both answers below are acceptable.

# Count the distinct UserNames.

SELECT COUNT(\*)

FROM (

# Get the UserNames and their review count (with more than one review).

SELECT UserName, COUNT(\*) AS CNT

FROM Reviews

GROUP BY UserName

# Note that MySQL allows reference to the alias.

# So this works too:

# HAVING CNT > 1

HAVING COUNT(\*) > 1) AS ReviewCnt;

# Alternatively (not preferred), filtering at the outer query.

SELECT COUNT(UserName)

FROM (

# Number of reviews per UserName. This will result

# in unique UserNames.

SELECT UserName, COUNT(\*) AS CNT

FROM Reviews

GROUP BY UserName) AS ReviewCnt

WHERE CNT > 1;

# 3. What day of the week is most popular for making a reservation?

# Use the DAYOFWEEK() function on the ‘Start’ column.

# Preferred (MySQL): Single-level SELECT statement.

SELECT DAYOFWEEK(Start) AS Day\_Of\_Week, COUNT(\*) AS Cnt

FROM Reservations

# MySQL allows grouping by the alias, Day\_Of\_Week.

GROUP BY DAYOFWEEK(Start)

ORDER BY Cnt DESC

LIMIT 1;

# Alternative (Standard SQL): Nested SELECT statement.

SELECT Day\_Of\_Week, COUNT(\*) AS Cnt

FROM (

# Day of week for each reservation.

SELECT ReservationId, DAYOFWEEK(Start) AS Day\_Of\_Week

FROM Reservations) AS ReservationsByDay

GROUP BY Day\_Of\_Week

ORDER BY Cnt DESC

LIMIT 1;

# 4. Which usernames have made multiple reservations at the same SitDownRestaurant?

# Note that one UserName can make multiple reservations

# at multiple restaurants. The outer "GROUP BY UserName" removes duplicate usernames.

# Alternatively, an outer SELECT DISTINCT UserName without the

# GROUP BY will yield the same results. I don't expect familiarity with this advanced keyword.

# It's essentially a GROUP BY anyway, so conceptually is equivalent to our answer.

# Similarly:

# SELECT DISTINCT UserName FROM Reservations GROUP BY UserName, RestaurantId HAVING COUNT(\*) > 1;

SELECT UserName

FROM (

# UserNames and RestaurantIds where the user has made more

# than one reservation at the restaurant.

SELECT UserName, RestaurantId, COUNT(\*) AS Cnt

FROM Reservations

GROUP BY UserName, RestaurantId

HAVING Cnt > 1) AS MultipleReservations

GROUP BY UserName;

# 5. Identify the number of credit cards per provider.

# The credit card provider is determined by the leading digit(s)

# of the CardNumber. Use the provided mapping for card providers.

# A nested IF(expr1,expr2,expr3) block would work, too,

# but the CASE block is easier to read.

# Note: can you express this without a nested SELECT statement?

# Try SUM(IF()).

SELECT Provider, COUNT(\*) AS Provider\_Cnt

FROM (

SELECT

CardNumber,

# CASE statement for mapping CardNumber to provider.

CASE

WHEN

LEFT(CAST(CardNumber AS CHAR),2) IN ('34','37')

THEN 'American Express'

WHEN

LEFT(CAST(CardNumber AS CHAR),4) IN ('6011') OR

LEFT(CAST(CardNumber AS CHAR),3) IN ('644','645','646','647','648','649') OR

LEFT(CAST(CardNumber AS CHAR),2) IN ('65')

THEN 'Discover'

WHEN

LEFT(CAST(CardNumber AS CHAR),2) IN ('51','52','53','54','55')

THEN 'Master Card'

WHEN

LEFT(CAST(CardNumber AS CHAR),1) IN ('4')

THEN 'Visa'

ELSE 'NA'

END AS Provider

FROM CreditCards) AS T

GROUP BY Provider;

# 6. What is the total number of active restaurants for each type of

# restaurant (SitDownRestaurant, TakeOutRestaurant, FoodCartRestaurant)?

SELECT 'SitDown' AS RestaurantType, COUNT(SitDownRestaurant.RestaurantId) AS ActiveRestaurants

FROM SitDownRestaurant

INNER JOIN Restaurants

ON SitDownRestaurant.RestaurantId = Restaurants.RestaurantId

WHERE Restaurants.Active = true

UNION

SELECT 'TakeOut' AS RestaurantType, COUNT(TakeOutRestaurant.RestaurantId) AS ActiveRestaurants

FROM TakeOutRestaurant

INNER JOIN Restaurants

ON TakeOutRestaurant.RestaurantId = Restaurants.RestaurantId

WHERE Restaurants.Active = true

UNION

SELECT 'FoodCart' AS RestaurantType, COUNT(FoodCartRestaurant.RestaurantId) AS ActiveRestaurants

FROM FoodCartRestaurant

INNER JOIN Restaurants

ON FoodCartRestaurant.RestaurantId = Restaurants.RestaurantId

WHERE Restaurants.Active = true;

# 7. Which UserNames have not created a review,

# nor created a recommendation, nor created a reservation?

SELECT Users.UserName

FROM Users

LEFT OUTER JOIN

Reviews

ON Users.UserName = Reviews.UserName

LEFT OUTER JOIN

Recommendations

ON Users.UserName = Recommendations.UserName

LEFT OUTER JOIN

Reservations

ON Users.UserName = Reservations.UserName

# Filter on whether the PK of the right-hand side table references are NULL.

# Filtering on the FKs would yield the same result. For example:

# WHERE Reviews.UserName IS NULL

# AND Recommendations.UserName IS NULL

# AND Reservations.UserName IS NULL

WHERE Reviews.ReviewId IS NULL

AND Recommendations.RecommendationId IS NULL

AND Reservations.ReservationId IS NULL;

# 8. What is the ratio of the total number of recommendations to the total number of reviews?

# CROSS JOIN is the simplest way to operate on a single value

# result from two incompatible table references.

SELECT (1.0\* RecommendationCount / ReviewCount) AS Recommendation\_Review\_Ratio

FROM

# Total number of recommendations.

(SELECT COUNT(\*) AS RecommendationCount

FROM Recommendations) AS Rec

CROSS JOIN

# Total number of reviews.

(SELECT COUNT(\*) AS ReviewCount

FROM Reviews) AS Rvw;

# Alternative (not preferred): INNER JOIN can be used as well, using an pseudo join key.

# Operationally, this is equivalent to the CROSS JOIN, but the

# SELECT statement is more complex.

SELECT (1.0\* RecommendationCount / ReviewCount) AS Recommendation\_Review\_Ratio

FROM

# Total number of recommendations.

(SELECT COUNT(\*) AS RecommendationCount, 1 AS JoinKey

FROM Recommendations) AS Rec

INNER JOIN

# Total number of reviews.

(SELECT COUNT(\*) AS ReviewCount, 1 AS JoinKey

FROM Reviews) AS Rvw

ON Rec.JoinKey = Rvw.JoinKey;

# Alternative (not preferred): both inner SELECT statements return a single numeric

# value, so we can divide the results inline. This is not standard SQL, but MySQL

# recognizes the single numeric value returned by each inner SELECT.

SELECT

(SELECT COUNT(\*) FROM Recommendations) /

(SELECT COUNT(\*) FROM Reviews);

# 9. Of the users that have created a reservation at a SitDownRestaurant,

# what is the percentage that the user has recommended that SitDownRestaurant?

# When there is no GROUP BY clause, the aggregation functions operate

# on the entire intermediate result set of the FROM-WHERE clauses.

SELECT 100.0 \* SUM(IF(RecommendationUsers.UserName IS NOT NULL, 1, 0))

/ COUNT(\*) AS Percent\_Reservation\_Recommendation

# To see the user reservations vs user recommendations, use:

# SELECT ReservationUsers.UserName, ReservationUsers.RestaurantId,

# RecommendationUsers.UserName, RecommendationUsers.RestaurantId

FROM

# Distinct UserNames and RestaurantIds for reservations on the left-hand side.

(SELECT UserName, RestaurantId

FROM Reservations

GROUP BY UserName, RestaurantId) AS ReservationUsers

LEFT OUTER JOIN

# Distinct UserNames AND RestaurantIds for recommended SitDownRestaurants on the right-hand side.

(SELECT UserName, Recommendations.RestaurantId AS RestaurantId

FROM Recommendations

INNER JOIN SitDownRestaurant

ON Recommendations.RestaurantId = SitDownRestaurant.RestaurantId

GROUP BY UserName, Recommendations.RestaurantId) AS RecommendationUsers

# Join on both the UserName and RestaurantId since we want to know

# the user recommended restaurants.

ON ReservationUsers.UserName = RecommendationUsers.UserName

AND ReservationUsers.RestaurantId = RecommendationUsers.RestaurantId;

# 10. Which UserNames have created more than twice the number of

# recommendations than number of reviews?

# 1. Identify all users.

SELECT \* FROM Users;

# 2. Recommendations per user.

SELECT UserName, COUNT(\*) AS Recommendation\_Cnt

FROM Recommendations

GROUP BY UserName;

# 3. Reviews per user.

SELECT UserName, COUNT(\*) AS Review\_Cnt

FROM Reviews

GROUP BY UserName;

# 4. View of the LEFT OUTER JOIN, which contains all users,

# even if they have zero recommendations or zero reviews.

SELECT Users.UserName,

UserRecommendations.UserName, UserRecommendations.Recommendation\_Cnt,

UserReviews.UserName, UserReviews.Review\_Cnt

FROM Users

LEFT OUTER JOIN

# Recommendations per user.

(SELECT UserName, COUNT(\*) AS Recommendation\_Cnt

FROM Recommendations

GROUP BY UserName) AS UserRecommendations

ON Users.UserName = UserRecommendations.UserName

LEFT OUTER JOIN

# Reviews per user.

(SELECT UserName, COUNT(\*) AS Review\_Cnt

FROM Reviews

GROUP BY UserName) AS UserReviews

ON Users.UserName = UserReviews.UserName;

# 5. Handle NULLs.

SELECT Users.UserName,

UserRecommendations.UserName,

IF(UserRecommendations.UserName IS NULL, 0, UserRecommendations.Recommendation\_Cnt) AS Recommendation\_Subtotal,

UserReviews.UserName,

IF(UserReviews.UserName IS NULL, 0, UserReviews.Review\_Cnt) AS Review\_Subtotal

FROM Users

LEFT OUTER JOIN

# Recommendations per user.

(SELECT UserName, COUNT(\*) AS Recommendation\_Cnt

FROM Recommendations

GROUP BY UserName) AS UserRecommendations

ON Users.UserName = UserRecommendations.UserName

LEFT OUTER JOIN

# Reviews per user.

(SELECT UserName, COUNT(\*) AS Review\_Cnt

FROM Reviews

GROUP BY UserName) AS UserReviews

ON Users.UserName = UserReviews.UserName;

# 6. Row filtering (selection) for outer-most query.

# A. The SELECT clause has a metric that is calculated after WHERE filtering.

# So try another outer query.

SELECT T.UserName, T.Recommendation\_Subtotal, T.Review\_Subtotal

FROM

(SELECT Users.UserName,

IF(UserRecommendations.UserName IS NULL, 0, UserRecommendations.Recommendation\_Cnt) AS Recommendation\_Subtotal,

IF(UserReviews.UserName IS NULL, 0, UserReviews.Review\_Cnt) AS Review\_Subtotal

FROM Users

LEFT OUTER JOIN

# Recommendations per user.

(SELECT UserName, COUNT(\*) AS Recommendation\_Cnt

FROM Recommendations

GROUP BY UserName) AS UserRecommendations

ON Users.UserName = UserRecommendations.UserName

LEFT OUTER JOIN

# Reviews per user.

(SELECT UserName, COUNT(\*) AS Review\_Cnt

FROM Reviews

GROUP BY UserName) AS UserReviews

ON Users.UserName = UserReviews.UserName) AS T

WHERE T.Recommendation\_Subtotal > 2 \* T.Review\_Subtotal;

# B. The SELECT aliases cannot be referenced in the WHERE clause.

# Recall WHERE filtering happens first.

# So duplicate the SELECT expr down into WHERE filtering.

SELECT Users.UserName,

IF(UserRecommendations.UserName IS NULL, 0, UserRecommendations.Recommendation\_Cnt) AS Recommendation\_Subtotal,

IF(UserReviews.UserName IS NULL, 0, UserReviews.Review\_Cnt) AS Review\_Subtotal

FROM Users

LEFT OUTER JOIN

# Recommendations per user.

(SELECT UserName, COUNT(\*) AS Recommendation\_Cnt

FROM Recommendations

GROUP BY UserName) AS UserRecommendations

ON Users.UserName = UserRecommendations.UserName

LEFT OUTER JOIN

# Reviews per user.

(SELECT UserName, COUNT(\*) AS Review\_Cnt

FROM Reviews

GROUP BY UserName) AS UserReviews

ON Users.UserName = UserReviews.UserName

WHERE IF(UserRecommendations.UserName IS NULL, 0, UserRecommendations.Recommendation\_Cnt)

> 2 \* IF(UserReviews.UserName IS NULL, 0, UserReviews.Review\_Cnt);