**Question 1:**

Write a query to find customers with at least one funded savings plan AND one

funded investment plan, sorted by total deposits.

**Answer 1:**

SELECT

pp.owner\_id,

concat(uc.first\_name, ' ', uc.last\_name) AS name,

COUNT(CASE WHEN pp.is\_regular\_savings = 1 AND sa.new\_balance > 0 THEN 1 END) AS savings\_count,

COUNT(CASE WHEN pp.is\_a\_fund = 1 AND sa.new\_balance > 0 THEN 1 END) AS investment\_count,

SUM(sa.confirmed\_amount) AS total\_deposits

FROM

plans\_plan AS pp

JOIN

savings\_savingsaccount AS sa ON sa.owner\_id = pp.owner\_id

JOIN

users\_customuser uc ON uc.id = pp.owner\_id

GROUP BY

pp.owner\_id, name

HAVING

savings\_count > 0 AND investment\_count > 0

ORDER BY

total\_deposits DESC;

**Explanation:**

There are three separate tables involved in answering this question (users\_customuser, savings\_savingsaccount, plans\_plan).

These three tables can be put into a single unit using JOIN. My choice of INNER JOIN is because we only need results with a match across the three tables i.e., users or customers whose ID appears across the three tables.

Using the AS statement, I renamed each of those tables to maintain a simpler and clearer query. Also, it is more efficient to write the alias rather than each table name every now and then.

The tables were joined based on the common columns

1. The common column between users\_customuser table and plans\_plan is owner\_id and id on the user table
2. The common column between users\_customuser and savings\_savingsaccount is owner\_id and id on user table also

Note: the alias of the three tables were used in the select statement because we need mysql to understand from what table are we choosing a certain column i.e. we have column “name” across all tables which would lead to error if we don’t specify what table we need that name column from.

CONCAT(uc.first\_name,' ', uc.last\_name) AS name: This would join the first and last name together and store under the name column

COUNT(CASE WHEN pp.is\_regular\_savings = 1 AND sa.new\_balance > 0 THEN 1 END) AS savings\_count: using the CASE statement which is like IF statement in Excel, we categorized customers with a funded savings account. After which we COUNT the number of customers that obey both conditions

COUNT(CASE WHEN pp.is\_a\_fund = 1 AND sa.new\_balance > 0 THEN 1 END) AS investment\_count: same as above. We categorize customers with a funded investment account and then COUNT it as investment\_count.

This way, mysql would only count rows that meets the conditions

The GROUP BY statement is to roll results into just one category i.e. if I have the same user\_id, I can roll it all up under one user\_id instead of many.

The SUM function is to get the total deposits made by the user(s) gotten from running the query. So, we roll the user\_id up (GROUP BY) because a user may have deposited on various days, so we need to put them all in one user\_id and then sum up the total deposit made by that user under that user\_id.

HAVING savings\_count > 0 AND investment\_count > 0: This works like WHERE statement except that it is for aggregated group of rows. This filters rows whose collective count is greater than 0

ORDER BY total\_deposits DESC: this orders the entire result based on the total deposits from highest to lowest