Assignment 3

Assignment Guideline: Creating Data for Database Tables

Objective:

This assignment aims to populate a set of database tables with data for testing purposes. This includes generating data for 13 users across multiple tables.

- 13 user entries, including username, full name, email, phone number, location, and job, were added to the user table.
 - Created data for the Account table, linking each user to an account.
 - Added categories to the category table for transaction classification purposes.
 - Generated transactions for each user, assigning random amounts and categories.
- Created goals for each user, including target amounts, completed amounts, target dates, and statuses.
 - Included reminders for bills with due dates and recurring status.
- 3. Additional information for existing users:
 - Expanded each user's data with additional transactions, goals, and reminders.

Assignment Steps:

To test my databases, I use queries with N joins for my tables, at least one aggregation function, and one group by/having clause.

1. Create queries to check all of John Doe's transactions and information in the table.

```
SELECT u.FullName,t.Date,c.CategoryName, SUM(t.Amount) AS TotalAmount FROM Transaction t

JOIN Category c ON t.CategoryID = c.CategoryID

JOIN Account a ON t.AccountID = a.AccountID

JOIN User u ON a.UserID = u.UserID

WHERE u.FullName = 'John Doe'

GROUP BY c.CategoryName,t.Date;
```

Queries:



2. Create queries to check Goals for all my users and their progress

Queries:

SELECT u.FullName,
SUM(g.AchievedAmount) AS TotalAchievedAmount,
SUM(g.TargetAmount) AS TotalTargetAmount
FROM User u
LEFT JOIN Account a ON u.UserID = a.UserID
LEFT JOIN Goal g ON a.AccountID = g.AccountID
GROUP BY u.FullName;

- We use SUM() to calculate the total target amount and total achieved amount for each user's goals.
- We use MAX() to determine the overall status of the user's goals. Since 'In Progress' is lexicographically greater than 'Not Started', 'Completed', etc., taking the maximum status will give us the overall status of the user's goals.
- We group the results by the full name of the user.

Result Grid

Result Grid Filter Rows:			
	FullName	TotalAchievedAmount	TotalTargetAmount
•	John Doe	1520.51	2300.10
	Jane Smith	1992.36	7251.52
	Alice Johnson	1753.26	7070.62
	Michael Johnson	2114.49	1841.03
	Emily Brown	1323.79	6356.15
	David Wilson	3369.67	5356.21
	Jennifer Lee	86.87	11871.38
	Ryan Garcia	3809.12	7720.80
	Sophia Martinez	2934.45	6429.40
	Ethan Anderson	2204.54	17205.49
	Olivia Rodriguez	1926.14	9750.86
	Noah Hernandez	657.11	3588.96
	Ava Lopez	1186.76	6262.44

3. To check how much money each user has spent on groceries.

Queries:

```
SELECT u.FullName, c.CategoryName, SUM(t.Amount) AS TotalSpentOnGroceries FROM User u

LEFT JOIN Account a ON u.UserID = a.UserID

LEFT JOIN Transaction t ON a.AccountID = t.AccountID

LEFT JOIN Category c ON t.CategoryID = c.CategoryID

WHERE c.CategoryName = 'Groceries'

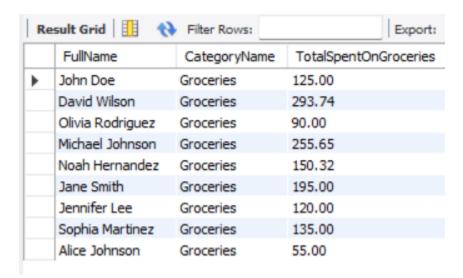
GROUP BY u.FullName, c.CategoryName

HAVING TotalSpentOnGroceries > 50;
```

in this query:

- We select the full name of each user from the user table.
- We use a LEFT JOIN to ensure that all users are included in the result set, even if they don't have any transactions associated with them.
- We use another LEFT JOIN to join the transaction table with the category table based on the category ID.
- We filter the transactions to only include those with the category name 'Groceries'.
- We sum up the amount spent on groceries by each user using the SUM() function.
- We group the results by the full name of the user.
- Having By to set minimum amount >50

This query will provide a list of all users along with the total amount of money they spent on groceries.



Each query contributes to understanding different aspects of the dataset and helps in analyzing user behavior and financial trends.